"P "VIS A VIS "NP" CONCATENATED WITH "NPC" AND CONCOMITANT "NP(HARD)"PROBLEM – AN AVANT GARDE AU COURANT MODEL A LA PETITO PRINCIPLI.

[1]DR K N PRASANNA KUMAR, [2]PROF B S KIRANAGI AND [3]PROF C S BAGEWADI

ABSTRACT**:* One long outstanding problem in mathematics and computer science is the P versus NP problem. While many may have heard of the P vs. NP problem in computational science through pop culture references (*The Simpsons*, *Futurama*) few understand its importance to modern computing, or what quantum computing may mean in relation to it. In computational complexity theory, P and NP are two classes of problems. P is the class of **decision problems** that a deterministic Turing machine can solve in polynomial time. Now, what that means in more useful terms is that any problem in P can be solved in less than $c*n^k$ steps where c and k are constants, independent of the input size, n. NP on the other hand, are problems that can be solved on ***non**deterministic Turing machines in polynomial time**. A *solution* to a NP problem can be verified on a deterministic Turing machine in polynomial time. The key difference is the machine type that is used, an NP problem cannot generally be solved in polynomial time on a deterministic Turing machine, it often has super-polynomial runtimes, e.g. $c*k^n$. Again, where c and k are constants independent of the input size, n. As an example of an easy to check, but hard to find solution look at the subset sum problem, determining whether or not a subset of numbers adds to zero is easy, but picking that subset from a large group is very difficult. A prime example of the difference between P and NP problems is that of finding Eulerian and Hamiltonian circuits on a given graph. A Eulerian circuit is a path around a graph that travels across each edge just once. It can be easily solved in polynomial time by checking for a graph's connectivity along with ensuring each vertex is connected to an even number of other vertices. The (very) closely related problem of find a Hamiltonian circuit—a path that touches each vertex just once—is not so simple and is an NP-complete problem. In order to fully solve this one would have to travel EVERY possible path (the number of paths increase exponentially with increasing number of vertices and nodes) until it either finds one that only touches each vertex once, or runs out of possible paths and determines that none exist. Here we assume two cases one where accentuates NP and the other P dissipates NP. AND NP (HARD)-VIS A VIS NPC. We assume the premises and see what the prediction results for both and NP are. Stability analysis, Solutional behaviour and Asymptotic Stability must all throw nevertheless locus and focus on the principal frontier of determinate apriori and differential posteori, ipso facto fait accompli desideratum.

INTRODUCTION:

Implications on P = NP

1.A common fallacious argument made is that since any quantum algorithm takes $\Omega(\sqrt{N})$ operations to identify a single marked element in a database of N elements, a quantum computer **cannot be used** to attain exponential speed up in a search problem. This argument is incorrect because this lower bound applies only to queries of the type used in Grover's algorithm, whose queries ask only about a single database element at a time. [Grover97]Various novel approaches can be used to get around the $\Omega(\sqrt{N})$ queries barrier which still leave hope for a finding some exponential speed up of an NP-Hard problem. These approaches generally try to capitalize on some structure of the problem at hand, which Grover's algorithm does not do at all. Grover provides an algorithm which will locate a single marked element in a N element database in exactly 1 query. It does however require $O(N \log N)$ pre and post processing time. While much slower in overall running time for the best classical and quantum algorithms for the same task, it does demonstrate that the $\Omega(\sqrt{N})$ query limit is not necessarily rule out exponential speed up of quantum computers in a search problem. [Grover97]It has also been shown that nonlinear quantum mechanics imply polynomial time solutions for NP-complete problems, however the same paper notes that: ``Such nonlinearity is purely hypothetical: all known experiments confirm the linearity of quantum mechanics to a high degree of accuracy'' [Abrams98]

Quantum Death match: PvNP

.

2. Sometimes people read these definitions and infer that all problems in P are easy, or conversely all in NP are difficult. This is an incorrect view, since these analysis look at worst case scenarios, it is entirely possible to have a NP problem run much faster for a P problem given similarly sized input values. For example, a problem in P, who has a runtime of n^{10000} or 1,000,000n, will take much longer to solve than a problem in NP who has a runtime of $2^{n/10,000}$ for all values of n up into the tens of thousands. There is still another level to this discussion that is critical when discussing P, NP, and potential ties to quantum computing. There is a subset of NP problems that are considered to be the hardest problems in NP; they are named **NP-complete**. An NP-complete problem is one who is in NP, but has the property that all other problems in NP can be transformed or reduced to it, in polynomial time. Now, here comes the bug: should ANY problem that has been shown to be NP-complete be shown to have a solution in polynomial time (e.g. can be solved deterministically without having to examine all possible choices) then ALL problems in NP can be solved in polynomial time!

3 Should someone prove that an NP-complete problem is solvable in polynomial time, this would imply that many of the problems at heart of today's online security measures—factoring very large composite integers—which rely on the problem not having a simple solution and not being cracked in a "reasonable" amount of time (where reasonable is often taken to be around the lifetime of the universe), could be cracked in one's lifetime. However, given the ramifications of this and the complete lack of a proof that P=NP—that is that there is no known NP-complete problem with a polynomial time solution—many computer scientists believe that P! =NP, and our on-line shopping are safe for some time to come.

5. What does this have to do with quantum computers? As John Timmer pointed out in his article, a quantum computer can exploit entanglement and the superposition of quantum states to examine many possible outcomes simultaneously. Does this mean that a large enough quantum computer could overcome any NP-complete problem in only one quantum step (or quantum polynomial time)? The answer is no, this is a common misconception. While it is not proven that quantum computers cannot solve NP-complete problems in polynomial time, it is generally accepted that they cannot.

6. What they can do, however is solve problem in another complexity set known as "bounded error, quantum, polynomial time" (BQP), the quantum analogue of "bounded error polynomial time" (BPP) in classical computing theory. In this problem set the answer to a complex problem can be obtained in polynomial time with a certain probability. That is if you run it once, you can be guaranteed that you have a 2/3 (or other arbitrary probability) chance of coming up with the correct result, subsequent runs can greatly increase your odds of getting the right answer. A traditional quantum computer carries out linear operations, such as multiplying a matrix by a vector. It is theorized that if a quantum computer could carry out *non-linear* operations, then it would be capable of solving NP problems in polynomial time, but it is predicted that such and a machine would violate other commonly held principles of quantum mechanics.

P versus NP problem

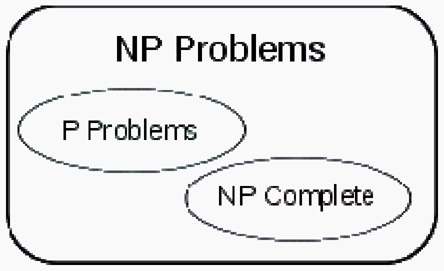


Diagram of complexity classes provided that **P**$\neq$ **NP**. The existence of problems within **NP** but outside both **P** and **NP**-complete, under that assumption, was established by Ladner's theorem.

7.The **P versus NP problem** is a major unsolved problem in computer science. Informally, it asks whether every problem whose solution can be quickly verified by a computer can also be quickly solved by a computer. It was introduced in 1971 by Stephen Cook in his seminal paper "The complexity of theorem proving procedures" and is considered by many to be the most important open problem in the field It is one of the seven Millennium Prize Problems selected by the Clay Mathematics Institute to carry a US$ 1,000,000 prize for the first correct solution The informal term *quickly* used above means the existence of an algorithm for the task that runs in polynomial time. The general class of questions for which some algorithm can provide an answer in polynomial time is called "class P" or just "**P**". For some questions, there is no known way to find an answer quickly, but if one is provided with information showing what the answer is, it may be possible to verify the answer quickly. The class of questions for which an answer can be *verified* in polynomial time is called **NP**.Consider the subset sum problem, an example of a problem that is easy to verify, but whose answer may be difficult to compute. Given a set of integers, does some nonempty subset of them sum to 0? For instance, does a subset of the set {−2, −3, 15, 14, 7, −10} add up to 0? The answer "yes, because {−2, −3, −10, 15} add up to zero" can be quickly verified with three additions. However, there is no known algorithm to find such a subset in polynomial time (there is one, however, in exponential time, which consists of 2^n-1 tries), and indeed such an algorithm cannot exist if the two complexity classes are not the same; hence this problem is in **NP** (quickly checkable) but not necessarily in **P** (quickly solvable).

8. An answer to the **P** = **NP** question would determine whether problems that can be verified in polynomial time, like the subset-sum problem, can also be solved in polynomial time. If it turned out that **P** does not equal **NP**, it would mean that there are problems in**NP** (such as NP-complete problems) that are harder to compute than to verify: they could not be solved in polynomial time, but the answer could be verified in polynomial time. Aside from being an important problem in computational theory, a proof either way would have profound implications for mathematics, cryptography, algorithm research, artificial intelligence, multimedia processing and many other fields.

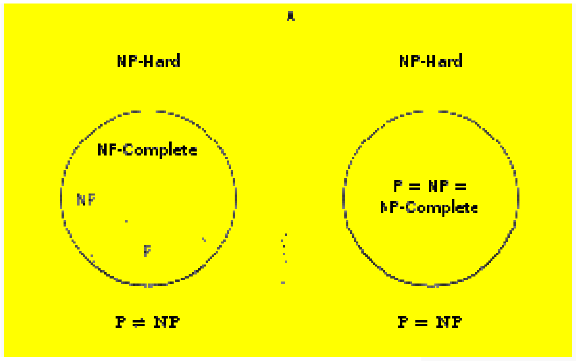
Context

9. The relation between the complexity classes **P** and **NP** is studied in computational complexity theory, the part of the theory of computation dealing with the resources required during computation to solve a given problem. The most common resources are time (how many steps it takes to solve a problem) and space (how much memory it takes to solve a problem).In such analysis, a model of the computer for which time must be analyzed is required. Typically such models assume that the computer is *deterministic* (given the computer's present state and any inputs, there is only one possible action that the computer might take) and *sequential* (it performs actions one after the other).In this theory, the class **P** consists of all those *decision problems* (defined below) that can be solved on a deterministic sequential machine in an amount of time that is polynomial in the size of the input; the class **NP** consists of all those decision problems whose positive solutions can be verified in polynomial time given the right information, or equivalently, whose solution can be found in polynomial time on a non-deterministic machine Clearly, **P** ⊆ **NP**. Arguably the biggest open question in theoretical computer science concerns the relationship between those two classes:

Is **P** equal to **NP**?

10.)In a 2002 poll of 100 researchers, 61 believed the answer to be no, 9 believed the answer is yes, and 22 were unsure; 8 believed the question may be independent of the currently accepted axioms and therefore is impossible to prove or disprove.

NP-complete



Euler diagram for P, NP, NP-complete, and NP-hard set of problems

To attack the **P** = **NP** question the concept of **NP**-completeness is very useful. **NP**-complete problems are a set of problems to each of which any other **NP**-problem can be reduced in polynomial time, and whose solution may still be verified in polynomial time. That is, any NP problem can be transformed into any of the NP-complete problems. Informally, an **NP**-complete problem is at least as "tough" as any other problem in **NP**.NP-hard problems are those at least as hard as **NP**-complete problems, i.e., all **NP**-problems can be reduced (in polynomial time) to them. **NP**-hard problems need not be in **NP**, i.e., they need not have solutions verifiable in polynomial time.For instance, the boolean satisfiability problem is **NP**-complete by the Cook-Levin theorem, so *any* instance of *any* problem in **NP** can be transformed mechanically into an instance of the boolean satisfiability problem in polynomial time. The Boolean satisfiability problem is one of many such **NP**-complete problems. If any **NP**-complete problem is in **P**, then it would follow that **P** = **NP**. Unfortunately, many important problems have been shown to be **NP**-complete, and as of 2012 not a single fast algorithm for any of them is known. Based on the definition alone it is not obvious that **NP**-complete problems exist. A trivial and contrived **NP**-complete problem can be formulated as: given a description of a Turing machine M guaranteed to halt in polynomial time, does there exist a polynomial-size input that M will accept? It is in **NP** because (given an input) it is simple to check whether M accepts the input by simulating M; it is **NP**-complete because the verifier for any particular instance of a problem in **NP** can be encoded as a polynomial-time machine M that takes the solution to be verified as input. Then the question of whether the instance is a yes or no instance is determined by whether a valid input exists. The first natural problem proven to be **NP**-complete was the Boolean satisfiability problem. As noted above, this is the Cook–Levin theorem; its proof that satisfiability is **NP**-complete contains technical details about Turing machines as they relate to the definition of **NP**. However, after this problem was proved to be **NP**-complete, reduction provided a simpler way to show that many other problems are also **NP**-complete, including the subset-sum problem discussed earlier. Thus, a vast class of seemingly unrelated problems is all reducible to one another, and is in a sense "the same problem".

Harder problems and Complexity class

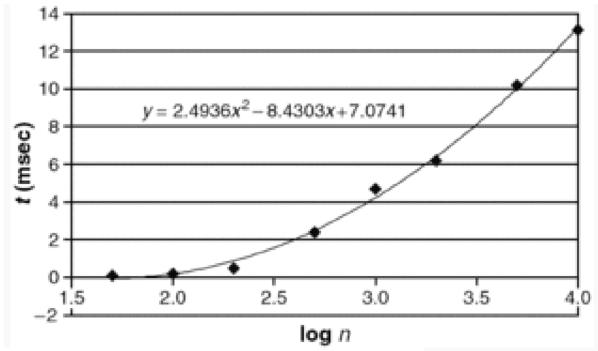
11) Although it is unknown whether **P** = **NP**, problems outside of **P** are known. A number of succinct problems (problems that operate not on normal input, but on a computational description of the input) are known to be **EXPTIME-complete**. Because it can be shown that **P** $\subsetneq$ **EXPTIME**, these problems are outside **P**, and so require more than polynomial time. In fact, by the time hierarchy theorem, they cannot be solved in significantly less than exponential time. Examples include finding a perfect strategy for chess (on an N×N board)[and some other board games.

The problem of deciding the truth of a statement in Presburger arithmetic requires even more time. Fischer and Rabin proved in 1974 that every algorithm that decides the truth of Presburger statements has a runtime of at least $2^{2^{cn}}$ for some constant *c*. Here, *n* is the length of the Presburger statement. Hence, the problem is known to need more than exponential run time. Even more difficult are the undecidable problems, such as the halting problem. They cannot be completely solved by any algorithm, in the sense that for any particular algorithm there is at least one input for which that algorithm will not produce the right answer; it will either produce the wrong answer, finish without giving a conclusive answer, or otherwise run forever without producing any answer at all.

Problems in NP not known to be in P or NP-complete

12) It was shown by Ladner that if **P** ≠ **NP** then there exist problems in **NP** that are neither in **P** nor **NP**-complete. Such problems are called NP-intermediate problems. The graph isomorphism problem, the discrete logarithm problem and the integer factorization problem are examples of problems believed to be **NP**-intermediate. They are some of the very few**NP** problems not known to be in **P** or to be **NP**-complete. The graph isomorphism problem is the computational problem of determining whether two finite graphs are isomorphic. An important unsolved problem in complexity theory is whether the graph isomorphism problem is in **P**, **NP**-complete, or **NP**-intermediate. The answer is not known, but it is believed that the problem is at least not **NP**-complete.[9] If graph isomorphism is **NP**-complete, the polynomial time hierarchy collapses to its second level. Since it is widely believed that the polynomial hierarchy does not collapse to any finite level, it is believed that graph isomorphism is not **NP**-complete. The best algorithm for this problem, due to Laszlo Babai and Eugene Luks has run time $2^{O(\sqrt{(n \log n)})}$ for graphs with n vertices.The integer factorization problem is the computational problem of determining the prime factorization of a given integer. Phrased as a decision problem, it is the problem of deciding whether the input has a factor less than k. No efficient integer factorization algorithm is known, and this fact forms the basis of several modern cryptographic systems, such as theRSA algorithm. The integer factorization problem is in **NP** and in **co-NP** (and even in UP and co-UP). If the problem is **NP**-complete, the polynomial time hierarchy will collapse to its first level (i.e., **NP** = **co-NP**). The best known algorithm for integer factorization is the general number field sieve, which takes expected time $O(e^{(64/9)1/3(n.\log 2)1/3(\log (n.\log 2))2/3})$ to factor an n-bit integer. However, the best known quantum algorithm for this problem, Shor's algorithm, does run in polynomial time. Unfortunately, this fact doesn't say much about where the problem lies with respect to non-quantum complexity classes.

Does P mean "easy"?



13) The graph shows time (average of 100 instances in msec using a 933 MHz Pentium III) vs. Problem size for knapsack problems for a state-of-the-art specialized algorithm. Quadratic fit suggests that empirical algorithmic complexity for instances with 50–10,000 variables is O $((\log n)^2)$. All of the above discussion has assumed that **P** means "easy" and "not in **P**" means "hard", an assumption known as*Cobham's thesis*. It is a common and reasonably accurate assumption in complexity theory, however it has some caveats. First, it is not always true in practice. A theoretical polynomial algorithm may have extremely large constant factors or exponents thus rendering it impractical. On the other hand, even if a problem is shown to be NP-complete, and even if **P** ≠**NP**, there may still be effective approaches to tackling the problem in practice. There are algorithms for many NP-complete problems, such as the knapsack problem, the traveling salesman problem and the Boolean satisfiability problem, that can solve to optimality many real-world instances in reasonable time. The empirical average-case complexity (time vs. problem size) of such algorithms can be surprisingly low. A famous example is the simplex algorithmin linear programming, which works surprisingly well in practice; despite having exponential worst-case time complexity it runs on par with the best known polynomial-time algorithms Second, there are types of computations which do not conform to the Turing machine model on which **P** and **NP** are defined, such as quantum computation and randomized algorithms.

Reasons to believe P ≠ NP

14) According to polls, many computer scientists believe that **P** ≠ **NP**. A key reason for this belief is that after decades of studying these problems no one has been able to find a polynomial-time algorithm for any of more than 3000 important known **NP**-complete problems (see List of NP-complete problems). These algorithms were sought long before the concept of **NP**-completeness was even defined (Karp's 21 NP-complete problems, among the first found, were all well-known existing problems at the time they were

shown to be NP-complete). Furthermore, the result **P** = **NP** would imply many other startling results that are currently believed to be false, such as **NP** = **co-NP** and **P** = **PH**

.It is also intuitively argued that the existence of problems that are hard to solve but for which the solutions are easy to verify matches real-world experience If P = NP, then the world would be a profoundly different place than we usually assume it to be. There would be no special value in "creative leaps," no fundamental gap between solving a problem and recognizing the solution once it's found. Everyone who could appreciate a symphony would be Mozart; everyone who could follow a step-by-step argument would be Gauss...

— Scott Aaronson, MIT

On the other hand, some researchers believe that there is overconfidence in believing **P** ≠ **NP** and that researchers should explore proofs of **P** = **NP** as well. For example, in 2002 these statements were made

The main argument in favor of **P** ≠ **NP** is the total lack of fundamental progress in the area of exhaustive search. This is, in my opinion, a very weak argument. The space of algorithms is very large and we are only at the beginning of its exploration. [. . .] The resolution of Fermat's Last Theorem also shows that very simple questions may be settled only by very deep theories.

—Moshe Y. Vardi, Rice University

Being attached to a speculation is not a good guide to research planning. One should **always try both directions of every problem**. Prejudice has caused famous mathematicians to fail to solve famous problems whose solution was opposite to their expectations, even though they had developed all the methods required.

—Anil Nerode, Cornell University

Consequences of the resolution of the problem

15)One of the reasons the problem attracts so much attention is the consequences of the answer. Either direction of resolution would advance theory enormously, and perhaps have huge practical consequences as well.

]**P** = **NP**

A proof that **P** = **NP** could have stunning practical consequences, if the proof leads to efficient methods for solving some of the important problems in **NP**. It is also possible that a proof would not lead directly to efficient methods, perhaps if the proof is non-constructive, or the size of the bounding polynomial is too big to be efficient in practice. The consequences, both positive and negative, arise since various **NP**-complete problems are fundamental in many fields.

Cryptography, for example, relies on certain problems being difficult. A constructive and efficient solution to an **NP**-complete problem such as 3-SAT would break most existing cryptosystems including public-key cryptography, a foundation for many modern security applications such as secure economic transactions over the Internet, and symmetric ciphers such as AES or 3DES, used for the encryption of communications data. These would need to be modified or replaced by information-theoretically secure solutions.

On the other hand, there are enormous positive consequences that would follow from rendering tractable many currently mathematically intractable problems. For instance, many problems in operations research are **NP**-complete, such as some types of integer programming, and the travelling salesman problem, to name two of the most famous examples. Efficient solutions to these problems would have enormous implications for logistics. Many other important problems, such as some problems in protein structure prediction, are also **NP**-complete;[if these problems were efficiently solvable it could spur considerable advances in biology.

But such changes may pale in significance compared to the revolution an efficient method for solving **NP**-complete problems would cause in mathematics itself. According to Stephen

...it would transform mathematics by allowing a computer to find a formal proof of any theorem which has a proof of a reasonable length, since formal proofs can easily be recognized in polynomial time. Example problems may well include all of the CMI prize problems.

Research mathematicians spend their careers trying to prove theorems, and some proofs have taken decades or even centuries to find after problems have been stated—for instance, Fermat's Last Theorem took over three centuries to prove. A method that is guaranteed to find proofs to theorems, should one exist of a "reasonable" size, would essentially end this struggle.

P ≠ NP

16) A proof that showed that **P ≠ NP** would lack the practical computational benefits of a proof that **P = NP**, but would nevertheless represent a very significant advance in computational complexity theory and provide guidance for future research. It would allow one to show in a formal way that many common problems cannot be solved efficiently, so that the attention of researchers can be focused on partial solutions or solutions to other problems. Due to widespread belief in **P ≠ NP**, much of this focusing of research has already taken place. Also **P ≠ NP** still leaves open the average-case complexity of hard problems in **NP**. For example, it is possible that SAT requires exponential time in the worst case, but that almost all randomly selected instances of it are efficiently solvable. Russell Impagliazzo has described five hypothetical "worlds" that could result from different possible resolutions to the average-case complexity question. These range from "Algorithmica", where **P = NP** and problems like SAT can be solved efficiently in all instances, to "Crypto mania", where **P ≠NP** and generating hard instances of problems outside **P** is easy, with three intermediate possibilities reflecting different possible distributions of difficulty over instances of **NP-hard** problems. The "world" where **P ≠ NP** but all problems in **NP** are tractable in the average case is called "Heuristica" in the paper. A Princeton University workshop in 2009 studied the status of the five worlds.

Results about difficulty of proof

17) Although the **P = NP**? Problem itself remains open, despite a million-dollar prize and a huge amount of dedicated research, efforts to solve the problem have led to several new techniques. In particular, some of the most fruitful research related to the **P = NP** problem has been in showing that existing proof techniques are not powerful enough to answer the question, thus suggesting that novel technical approaches are required. As additional evidence for the difficulty of the problem, essentially all known proof techniques in computational complexity theory fall into one of the following classifications, each of which is known to be insufficient to prove that **P ≠ NP**:

Classification	**Definition**
Relativizing proofs	Imagine a world where every algorithm is allowed to make queries to some fixed subroutine called an oracle, and the running time of the oracle is not counted against the running time of the algorithm. Most proofs (especially classical ones) apply uniformly in a world with oracles regardless of what the oracle does. These proofs are called *relativizing*. In 1975, Baker, Gill, and Solovay showed that **P = NP** with respect to some oracles, while **P ≠ NP** for other oracles Since relativizing proofs can only prove statements that are uniformly true with respect to all possible oracles, this showed that relativizing techniques cannot resolve **P= NP**.
Natural proofs	In 1993, Alexander Razborov and Steven Rudich defined a general class of proof techniques for circuit complexity lower bounds, called *natural proofs*. At the time all previously known circuit lower bounds were natural, and circuit complexity was considered a very promising approach for resolving **P = NP**. However, Razborov and Rudich showed that, if one-way functions exist, then no natural proof method can distinguish between **P** and **NP**. Although one-way functions have never been formally proven to exist, most mathematicians believe that they do, and a proof or disproof of their existence would be a much stronger statement than the quantification of **P** relative to **NP**. Thus it is unlikely that natural proofs alone can resolve **P = NP**.

Algebrizing proofs	After the Baker-Gill-Solovay result, new non-relativizing proof techniques were successfully used to prove that IP = PSPACE. However, in 2008, Scott Aaronsonand Avi Wigderson showed that the main technical tool used in the **IP** = **PSPACE** proof, known as *arithmetization*, was also insufficient to resolve **P** = **NP**.

These barriers are another reason why **NP**-complete problems are useful: if a polynomial-time algorithm can be demonstrated for an **NP**-complete problem, this would solve the **P** =**NP** problem in a way not excluded by the above results.

These barriers have also led some computer scientists to suggest that the P versus NP problem may be independent of standard axiom systems like ZFC (cannot be proved or disproved within them). The interpretation of an independence result could be that either no polynomial-time algorithm exists for any NP-complete problem, and such a proof cannot be constructed in (e.g.) ZFC, or that polynomial-time algorithms for NP-complete problems may exist, but it's impossible to prove in ZFC that such algorithms are correct. However, if it can be shown, using techniques of the sort that are currently known to be applicable, that the problem cannot be decided even with much weaker assumptions extending the Peano axioms (PA) for integer arithmetic, then there would necessarily exist nearly-polynomial-time algorithms for every problem in NP. Therefore, if one believes (as most complexity theorists do) that not all problems in NP have efficient algorithms, it would follow that proofs of independence using those techniques cannot be possible. Additionally, this result implies that proving independence from PA or ZFC using currently known techniques is no easier than proving the existence of efficient algorithms for all problems in NP.

Claimed solutions

18) While the P versus NP problem is generally considered unsolved,[27] many amateur and some professional researchers have claimed solutions. Woeginger (2010) has a comprehensive list. An August 2010 claim of proof that P ≠ NP, by Vinay Deolalikar, researcher at HP Labs, Palo Alto, received heavy Internet and press attention after being initially described as "seem[ing] to be a relatively serious attempt" by two leading specialists.[The proof has been reviewed publicly by academics,[and Neil Immerman, an expert in the field, had pointed out two possibly fatal errors in the proof As of 15 September 2010, Deolalikar was reported to be working on a detailed expansion of his attempted proof However, opinions expressed by several notable theoretical computer scientists indicate that the attempted proof is neither correct nor a significant advancement in our understanding of the problem.[34]

Logical characterizations

19) The **P** = **NP** problem can be restated in terms of expressible certain classes of logical statements, as a result of work in descriptive complexity. All languages (of finite structures with a fixed signature including a linear order relation) in **P** can be expressed in first-order logic with the addition of a suitable least fixed-point combinatory (effectively, this, in combination with the order, allows the definition of recursive functions); indeed, (as long as the signature contains at least one predicate or function in addition to the distinguished order relation [so that the amount of space taken to store such finite structures is actually polynomial in the number of elements in the structure]), this precisely characterizes **P**. Similarly, **NP** is the set of languages expressible in existential second-order logic—that is, second-order logic restricted to exclude universal quantification over relations, functions, and subsets. The languages in the polynomial hierarchy, **PH**, correspond to all of second-order logic. Thus, the question "is **P** a proper subset of **NP**" can be reformulated as "is existential second-order logic able to describe languages (of finite linearly ordered structures with nontrivial signature) that first-order logic with least fixed point cannot?" The word "existential" can even be dropped from the previous characterization, since **P** = **NP** if and only if **P** = **PH** (as the former would establish that **NP** = **co-NP**, which in turn implies that **NP** = **PH**). PSPACE = NPSPACE as established Savitch's theorem, this follows directly from the fact that the square of a polynomial function is still a polynomial function. However, it is believed, but not proven, that a similar relationship may not exist between the polynomial time complexity classes P and NP, so the question is still open.

Polynomial-time algorithms

20)No algorithm for any **NP**-complete problem is known to run in polynomial time. However, there are algorithms for **NP**-complete problems with the property that if **P** = **NP**, then the algorithm runs in polynomial time (although with enormous constants, making the algorithm impractical). The following algorithm, due to Levin is such an example. It correctly accepts the **NP**-complete language SUBSET-SUM. It runs polynomial time if and only if **P** = **NP**:

```
// Algorithm that accepts the NP-complete language SUBSET-SUM.
//
// This is a polynomial-time algorithm if and only if P=NP.
//
// "Polynomial-time" means it returns "yes" in polynomial time when
// the answer should be "yes", and runs forever when it is "no".
//
// Input: S = a finite set of integers
// Output: "yes" if any subset of S adds up to 0.
// Runs forever with no output otherwise.
// Note: "Program number P" is the program obtained by
// writing the integer P in binary, then
// considering that string of bits to be a
// program. Every possible program can be
// generated this way, though most do nothing
//          because          of          syntax          errors.
FOR N = 1...infinity
FOR P = 1...N
Run program number P for N steps with input S
 IF the program outputs a list of distinct integers
  AND the integers are all in S
  AND          the          integers          sum          to          0
THEN
  OUTPUT "yes" and HALT
```

If, and only if, **P** = **NP**, then this is a polynomial-time algorithm accepting an **NP**-complete language. "Accepting" means it gives "yes" answers in polynomial time, but is allowed to run forever when the answer is "no".

This algorithm is enormously impractical, even if **P** = **NP**. If the shortest program that can solve SUBSET-SUM in polynomial time is *b* bits long, the above algorithm will try at least 2^b-1 other programs first.

Formal definitions for P and NP

21) Conceptually a *decision problem* is a problem that takes as input some string *w* over an alphabet Σ, and outputs "yes" or "no". If there is an algorithm (say a Turing machine, or a computer program with unbounded memory) that can produce the correct answer for any input string of length *n* in at most $c \cdot n^k$ steps, where *k* and *c* are constants independent of the input string, then we say that the problem can be solved in *polynomial time* and we place it in the class **P**. Formally, **P** is defined as the set of all languages that can be decided by a deterministic polynomial-time Turing machine. That is,

$$\mathbf{P} = \{L : L = L(M) \text{ for some deterministic polynomial-time Turing } $$

where

$$L(M) = \{w \in \Sigma^* : M \text{ accepts } w\}$$

and a deterministic polynomial-time Turing machine is a deterministic Turing machine *M* that satisfies the following two conditions:

M halts on all input *w* and

there exists $k \in N$ such that $T_M(n) \in O(n^k)$ (where O refers to the big O notation),

$$T_M(n) = \max\{t_M(w) : w \in \Sigma^*, |w| = n\}$$

and

NP can be defined similarly using nondeterministic Turing machines (the traditional way). However, a modern approach to define **NP** is to use the concept of *certificate* and *verifier*. Formally, **NP** is defined as the set of languages over a finite alphabet that have a verifier that runs in polynomial time, where the notion of "verifier" is defined as follows.

Let *L* be a language over a finite alphabet, Σ.

L ∈ **NP** if, and only if, there exists a binary relation $R \subset \Sigma^* \times \Sigma^*$ and a positive For all $x \in \Sigma^*$, $x \in L \Leftrightarrow \exists y \in \Sigma^*$ such that $(x, y) \in R$ and $|y| \in O(|x|^k)$; and

the language $L_R = \{x\#y : (x, y) \in R\}$ over $\Sigma \cup \{\#\}$ is decidable by a Turing machine in polynomial time.

A Turing machine that decides L_R is called a *verifier* for *L* and a *y* such that $(x, y) \in R$ is called a *certificate of membership* of *x* in *L*.

In general, a verifier does not have to be polynomial-time. However, for *L* to be in **NP**, there must be a verifier that runs in polynomial time.

Example

Let

$$COMPOSITE = \{x \in \mathbb{N} | x = pq \text{ for integers } p, q > 1\}$$

Clearly, the question of whether a given *x* is a composite is equivalent to the question of whether *x* is a member of COMPOSITE. It can be shown that COMPOSITE ∈ **NP** by verifying that it satisfies the above definition (if we identify natural numbers with their binary representations).

COMPOSITE also happens to be in **P**.

Formal definition for NP-completeness

There are many equivalent ways of describing **NP**-completeness. Let L be a language over a finite alphabet Σ.

s **NP**-complete if, and only if, the following two conditions are satisfied:

$L \in \mathbf{NP}$; and

any $L' \in \mathbf{NP}$ is polynomial-time-reducible to L (written as $L' \leq_p L$), where $L' \leq_p L$ if, and only if, the following two conditions are satisfied:

There exists $f : \Sigma^* \rightarrow \Sigma^*$ such that $\forall w \in \Sigma^* (w \in L' \Leftrightarrow f(w) \in L)$; and

there exists a polynomial-time Turing machine that halts with $f(w)$ on its tape on any input w.

Popular culture

Travelling Salesman (2012 film), by director Timothy Lanzone, is the story of 4 mathematicians hired by the US Government to solve the most elusive problem in computer-science history: P vs. NP

A SIMPLE EXAMPLE FOR CLARIFICATION:

22) Suppose that you are organizing housing accommodations for a group of four hundred university students. Space is limited and only one hundred of the students will receive places in the dormitory. To complicate matters, the Dean has provided you with a list of pairs of incompatible students, and requested that no pair from this list appear in your final choice. This is an example of what computer scientists call

an NP-problem, since it is easy to check if a given choice of one hundred students proposed by a coworker is satisfactory (i.e., no pair taken from your coworker's list also appears on the list from the Dean's office), however the task of generating such a list from scratch seems to be so hard as to be completely impractical. Indeed, the total number of ways of choosing one hundred students from the four hundred applicants is greater than the number of atoms in the known universe! Thus no future civilization could ever hope to build a supercomputer capable of solving the problem by brute force; that is, by checking every possible combination of 100 students. However, this apparent difficulty may only reflect the lack of ingenuity of your programmer. In fact, one of the outstanding problems in **computer science is determining whether** questions exist whose answer can be quickly checked, but which require an impossibly long time to solve by any direct procedure. Problems like the one listed above certainly seem to be of this kind, but so far no one has managed to prove that any of them really are so hard as they appear, i.e., that there really is no feasible way to generate an answer with the help of a computer. Stephen Cook and Leonid Levin formulated the P (i.e., easy to find) versus NP (i.e., easy to check) problem independently in 1971.

A STATUS QUO ANTE OF THE PROBLEM OF P AND NP(EXCERPTS FROM ARTICLE BY Lance Fortnow)

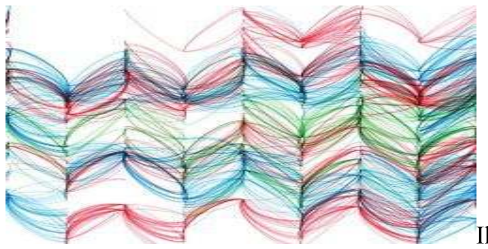


Illustration by C.E.B. Reas

23) Many believed that the quickly developing area of circuit complexity would soon settle the **P** versus **NP** problem, whether every algorithmic problem with efficiently verifiable solutions have efficiently computable solutions. But circuit complexity and other approaches to the problem have stalled and we have little reason to believe we will see a proof separating **P** from **NP** in the near future. Nevertheless, the computer science landscape has dramatically changed in the nearly four decades since Steve Cook presented his seminal **NP**-completeness paper "**The Complexity of Theorem-Proving Procedures" in Shaker Heights**, OH in early May, 1971. Computational power has dramatically increased; the cost of computing has dramatically decreased, not to mention the power of the Internet. Computation has become a standard tool in just about every academic field. Whole subfields of biology, chemistry, physics, economics and others are devoted to large-scale computational modeling, simulations, and problem solving. As we solve larger and more complex problems with greater computational power and cleverer algorithms, the problems we cannot tackle begin to stand out. The theory of **NP**-completeness helps us understand these limitations and the **P** versus **NP** problem begins to loom large not just as an interesting theoretical question in computer science, but as a basic principle that permeates all the sciences.So while we don't expect the **P** versus **NP** problem to be resolved in the near future, the question has driven research in a number of topics to help us understand, handle, and even take advantage of the hardness of various computational problems. Many people have tried to solve the **P** versus **NP** problem as well as how this question has shaped so much of the research in computer science and beyond. To handle **NP**-complete problems and the theory that has developed from those approaches. , it is necessary to have a new type of **"interactive proof systems"** led to limitations of approximation algorithms and consider whether quantum computing can solve **NP**-complete problems (short answer: not likely). To separate **P** from **NP** using algebraic-geometric techniques. Has been goal post of many a mathematician.

What is the P versus NP Problem?

24) Suppose we have a large group of students that we need to pair up to work on projects. We know

which students are compatible with each other and we want to put them in compatible groups of two. We could search all possible pairings but even for 40 students we would have more than 300 billion trillion possible pairings. In 1965, Jack Edmonds gave an efficient algorithm to solve this matching problem and suggested a formal definition of "efficient computation" (runs in time a fixed polynomial of the input size). The class of problems with efficient solutions would later become known as **P** for "Polynomial Time."But many related problems do not seem to have such an efficient algorithm. What if we wanted to make groups of three students with each pair of students in each group compatible (Partition into Triangles)? What if we wanted to find a large group of students all of whom are compatible with each other (Clique)? What if we wanted to sit students around a large round table with no incompatible students sitting next to each other (Hamiltonian Cycle)? What if we put the students into three groups so that each student is in the same group with only his or her compatibles (3-Coloring)?

25) All these problems have a similar favor: Given a potential solution, for example, a seating chart for the round table, we can validate that solution efficiently. The collection of problems that have efficiently verifiable solutions is known as **NP** (for "Nondeterministic Polynomial-Time," if you have to ask).So **P** = **NP** means that for every problem that has an efficiently verifiable solution, we can find that solution efficiently as well. We call the very hardest **NP** problems (which include Partition into Triangles, Clique, Hamiltonian Cycle and 3-Coloring) "NP-complete," that is, given an efficient algorithm for one of them, we can find an efficient algorithm for all of them and in fact any problem in **NP**. Steve Cook, Leonid Levin, and Richard Karp developed the initial theory of **NP**-completeness that generated multiple ACM Turing Awards. In the 1970s, theoretical computer scientists showed hundreds more problems **NP**-complete (see Garey and Johnson). An efficient solution to any **NP**-complete problem would imply **P** = **NP** and an efficient solution to every **NP**-complete problem. Most computer scientists quickly came to believe **P** ≠ **NP** and trying to prove it quickly became the single most important question in all of theoretical computer science and one of the most important in all of mathematics. Soon the **P** versus **NP** problem became an important computational issue in nearly every scientific discipline.
As computers grew cheaper and more powerful, computation started playing a major role in nearly every academic field, especially the sciences. The more scientists can do with computers, the more they realize some problems seem computationally difficult. Many of these fundamental problems turn out to be **NP**-complete. A small sample:

- **Finding a DNA sequence that best fits a collection of fragments of the sequence (see Gusfield).**
- **Finding a ground state in the Ising model of phase transitions (see Cipra).**
- **Finding Nash Equilibriums with specific properties in a number of environments (see Conitzer).**
- **Finding optimal protein threading procedures**.
- Determining if a mathematical statement has a short proof (follows from Cook).

In 2000, the Clay Math Institute named the **P** versus **NP** problem as one of the seven most important open questions in mathematics and has offered a million-dollar prize for a proof that determines whether or not **P** =**NP**.

What If P = NP?

To understand the importance of the **P** versus **NP** problem let us imagine a world where **P** = **NP**. Technically we could have **P** = **NP**, but not have practical algorithms for most **NP**-complete problems. But suppose in fact we do have very quick algorithms for all these problems. Many focus on the negative, that if **P** = **NP** then public-key cryptography becomes impossible. True, but what we will gain from **P** = **NP** will make the whole Internet look like a footnote in history.

What we would gain from P = NP will make the whole Internet look like a footnote in history.

26) Since all the **NP**-complete optimization problems become easy, everything will be much more efficient. Transportation of all forms will be scheduled optimally to move people and goods around quicker and cheaper. Manufacturers can improve their production to increase speed and create less waste. And I'm just scratching the surface. Learning becomes easy by using the principle of Occam's razor—we simply find the smallest program consistent with the data. Near perfect vision recognition, language comprehension and translation and all other learning tasks become trivial. We will also have much better predictions of weather and earthquakes and other natural phenomenon = **NP** would also have big implications in mathematics. One could find short, fully logical proofs for theorems but these proofs are

usually extremely long. But we can use the Occam razor principle to recognize and verify mathematical proofs as typically written in journals. Complexity theorists generally believe **P ≠ NP** and such a beautiful world cannot exist.

Approaches to Showing P ≠ NP

Diagonalization.
27) Can we just construct an **NP** language *L* specifically designed so that every single polynomial-time algorithm fails to compute *L* properly on some input? This approach, known as diagonalization, goes back to the 19th century.
In 1874, Georg Cantor showed the real numbers are uncountable using a technique known as diagonalization. Given a countable list of reals, Cantor showed how to create a new real number not on that list. Alan Turing, in his seminal paper on computation, used a similar technique to show that the Halting problem is not computable. In the 1960s complexity theorists used diagonalization to show that given more time or memory one can solve more problems. Why not use diagonalization to separate **NP** from **P**?
Diagonalization requires simulation and we don't know how a fixed **NP** machine can simulate an arbitrary **P**machine. Also a diagonalization proof would likely relativize, that is, work even if all machines involved have access to the same additional information. Baker, Gill and Solovay showed no relativizable proof can settle the **P** versus **NP** problem in either direction.
Complexity theorists have used diagonalization techniques to show some **NP**-complete problems like Boolean formula satisfiability cannot have algorithms that use both a small amount of time and memory, ^{b}ut this is a long way from **P ≠ NP**.
Circuit Complexity.
28) To show **P ≠ NP** it is sufficient to show some -complete problem cannot be solved by relatively small circuits of AND, OR, and NOT gates (the number of gates bounded by a fixed polynomial in the input size).
In 1984, Furst, Saxe, and Sipser showed that small circuits cannot solve the parity function if the circuits have a fixed number of layers of gates. In 1985, Razborov showed the **NP**-complete problem of finding a large clique does not have small circuits if one only allows AND OR gates (no NOT gates). If one extends Razborov's result to general circuits one will have proved **P ≠ NP**.
Razborov later showed his techniques would fail miserably if one allows NOT gates. Razborov and Rudich develop a notion of "natural" proofs and give evidence that our limited techniques in circuit complexity cannot be pushed much further. And, in fact, we haven't seen any significantly new circuit lower bounds in the past 20 years.
Proof Complexity.
29)Consider the set of Tautologies, the Boolean formulas θ of variables over ANDs, ORs, and NOTs such that every setting of the variables to True and False makes θ true, for example the formula

$(x \text{ AND } y) \text{ OR } (\text{NOT } x) \text{ OR } (\text{NOT } y).$

A literal is a variable or its negation, such as *x* or NOT *x*. A formula, like the one here, is in Disjunctive Normal Form (DNF) if it is the OR of ANDs of one or more literals.
If a formula θ is not a tautology, we can give an easy proof of that fact by exhibiting an assignment of the variables that makes θ false. But if θ were indeed a tautology, we don't expect short proofs. If one could prove there are no short proofs of tautology that would imply **P ≠ NP**.
Resolution is a standard approach to proving tautologies of DNFs by finding two clauses of the form (ψ_1 and) and (ψ_2 AND NOT *x*) and adding the clause (ψ_1 AND ψ_2). A formula is a tautology exactly when one can produce an empty clause in this manner.
In 1985, Wolfgang Haken showed that tautologies that encode the pigeonhole principle (*n* + 1 pigeons in *n*holes means some hole has more than one pigeon) do not have short resolution proofs.

Since then complexity theorists have shown similar weaknesses in a number of other proof systems including cutting planes, algebraic proof systems based on polynomials, and restricted versions of proofs using the Frege axioms, the basic axioms one learns in an introductory logic course. But to prove **P ≠ NP** we would need to show that tautologies cannot have short proofs in an arbitrary proof system. Even a breakthrough result showing tautologies don't have short general Frege proofs would not suffice in separating **NP** from **P**.

Dealing with Hardness

30) So you have an **NP**-complete problem you just have to solve. If, as we believe, **P ≠ NP** you won't find

a general algorithm that will correctly and accurately solve your problem all the time. But sometimes you need to solve the problem anyway. All hope is not lost. Here, I describe some of the tools one can use on **NP**-complete problems and how computational complexity theory studies these approaches. Typically one needs to combine several of these approaches when tackling **NP**-complete problems in the real world.
Brute Force.
31) Computers have gotten faster, much faster since **NP**-completeness was first developed. Brute force search through all possibilities is now possible for some small problem instances. With some clever algorithms we can even solve some moderate size problems with ease. The **NP**-complete traveling salesperson problem asks for the smallest distance tour through a set of specified cities. Using extensions of the cutting-plane method we can now solve, in practice, traveling salespeople problems with more than 10,000 cities .
Consider the 3SAT problem, solving Boolean formula satisfiability where formulas are in the form of the AND of several clauses where each clause is the OR of three literal variables or negations of variables). 3SAT remains **NP**-complete but the best algorithms can in practice solve SAT problems on about 100 variables. We have similar results for other variations of satisfiability and many other **NP**-complete problems.
But for satisfiability on general formulae and on many other **NP**-complete problems we do not know algorithms better than essentially searching all the possibilities. In addition, all these algorithms have exponential growth in their running times, so even a small increase in the problem size can kill what was an efficient algorithm. Brute force alone will not solve **NP**-complete problems no matter how clever we are.
*32) **Parameterized Complexity***.

Consider the Vertex Cover problem; find a set of *k* "central people" such that for every compatible pair of people, at least one of them is central. For small *k* we can determine whether a central set of people exists efficiently no matter the total number *n* of people we are considering. For the Clique problem even for small *k* the problem can still be difficult.Downey and Fellows developed a theory of parameterized complexity that gives a fine-grained analysis of the complexity of **NP**-complete problems based on their parameter size.
*33) **Approximation.***

We cannot hope to solve **NP**-complete optimization problems exactly but often we can get a good approximate answer. Consider the traveling salesperson problem again with distances between cities given as the crow flies (Euclidean distance). This problem remains **NP**-complete but Arora gives an efficient algorithm that gets very close to the best possible route.Consider the MAX-CUT problem of dividing people into two groups to maximize the number of incompatibles between the groups. Goemans and Williamson use semi-definite programming to give a division of people only a .878567 factor of the best possible.
*34) **Heuristics and Average-Case Complexity***.
The study of **NP**-completeness focuses on how algorithms perform on the worst possible inputs. However the specific problems that arise in practice may be much easier to solve. Many computer scientists employ various heuristics to solve **NP**-complete problems that arise from the specific problems in their fields.
While we create heuristics for many of the **NP**-complete problems, Boolean formula Satisfiability (SAT) receives more attention than any other. Boolean formulas, especially those in conjunctive normal form (CNF), the AND of ORs of variables and their negations, have a very simple description and yet are general enough to apply to a large number of practical scenarios particularly in software verification and artificial intelligence. Most natural **NP**-complete problems have simple efficient reductions to the satisfiability of Boolean formulas. In competition these SAT solvers can often settle satisfiability of formulas of one million variables Computational complexity theorists study heuristics by considering average-case complexity—how well can algorithms perform on average from instances generated by some specific distribution.
Leonid Levin developed a theory of efficient algorithms over a specific distribution and formulated a distributional version of the **P** versus **NP** problem.
Some problems, like versions of the shortest vector problem in a lattice or computing the permanent of a matrix, are hard on average exactly when they are hard on worst-case inputs, but neither of these problems is believed to be **NP**-complete. Whether similar worst-to-average reductions hold for **NP**-complete sets is an important open problem. Average-case complexity plays an important role in many areas of computer science, particularly cryptography, as discussed later.

35) **Interactive Proofs and Limits of Approximation**

Previously, we saw how sometimes one can get good approximate solutions to **NP**-complete optimization problems. Many times though we seem to hit a limit on our ability to even get good approximations. We now know that we cannot achieve better approximations on many of these problems unless **P** = **NP** and we could solve these problems exactly. The techniques to show these negative results came out of a new model of proof system originally developed for cryptography and to classify group theoretic algorithmic problems. As mentioned earlier, we don't expect to have short traditional proofs of tautologies. But consider an "interactive proof" model where a prover Peggy tries to convince a verifier Victor that a formula θ is a tautology. Victor can ask Peggy randomly generated questions and need only be convinced with high confidence. Quite surprisingly, these proof systems have been shown to exist not only for tautologies but for any problem computable in a reasonable amount of memory.
A variation known as a "probabilistically checkable proof system" (PCPs), where Peggy writes down an encoded proof and Victor can make randomized queries to the bits of the proof, has applications for approximations. The "PCP Theorem" optimizes parameters, which in its strong form shows that every language in **NP** has a PCP where Victor uses a tiny number of random coins and queries only three bits of the proof.
One can use this PCP theorem to show the limitations of approximation for a large number of optimization questions. For example, one cannot approximate the largest clique in a group of *n* people by more than a multiplicative ratio of nearly $\sqrt{n}$ unless **P** = **NP**. See Madhu Sudan's recent article in *Communications* for more details and references on PCPs. One can do even better assuming a "Unique Games Conjecture" that there exists PCPs for **NP** problems with some stronger properties. Consider the MAX-CUT problem of dividing people discussed earlier. If the unique games conjecture holds one cannot do better than the .878567 factor given by the Goemans-Williamson approximation algorithm. Recent work shows how to get a provably best approximation for essentially any constrained problem assuming this conjecture.

We expect P ≠ NP to hold in very strong ways. We can use strong hardness assumptions as a positive tool, particularly to create cryptographic protocols and to reduce or even eliminate the need of random bits in probabilistic algorithms.

Using Hardness

In "What If **P** = **NP**?" we saw the nice world that arises when we assume **P** = **NP**. But we expect **P** ≠ **NP** to hold in very strong ways. We can use strong hardness assumptions as a positive tool, particularly to create cryptographic protocols and to reduce or even eliminate the need of random bits in probabilistic algorithms. *Cryptography*. We take it for granted these days, the little key or lock on our Web page that tells us that someone listening to the network won't get the credit card number I just sent to an online store or the password to the bank that controls my money. But public-key cryptography, the ability to send secure messages between two parties that have never privately exchanged keys, is a relatively new development based on hardness assumptions of computational problems.If **P** = **NP** then public-key cryptography is impossible. Assuming **P** ≠ **NP** is not enough to get public-key protocols; instead we need strong average-case assumptions about the difficulty of factoring or related problems.
We can do much more than just public-key cryptography using hard problems. Suppose Alice's husband Bob is working on a Sudoku puzzle and Alice claims she has a solution to the puzzle (solving a $n \times n$ Sudoku puzzle is **NP**-complete). Can Alice convince Bob that she knows a solution without revealing any piece of it? Alice can use a "zero-knowledge proof," an interactive proof with the additional feature that the verifier learns nothing other than some property holds, like a Sudoku puzzle having a solution. Every **NP** search problem has a zero-knowledge proof under the appropriate hardness assumptions. Online poker is generally played through some "trusted" Web site, usually somewhere in the Caribbean. Can we play poker over the Internet without a trusted server? Using the right cryptographic assumptions, not only poker but any protocol that uses a trusted party can be replaced by one that uses no trusted party and the players can't cheat or learn anything new beyond what they could do with the trusted party.[b]
36) ***Eliminating Randomness.***

In the 1970s we saw a new type of algorithm, one that used random bits to aid in finding a solution to a problem. Most notably we had probabilistic algorithms[35] for determining whether a number is prime, an important routine needed for modern cryptography. In 2004, we discovered we don't need randomness at all to efficiently determine if a number is prime.2 Does randomness help us at all in finding solutions to **NP** problems? Truly independent and uniform random bits are either very difficult or impossible to produce (depending on your beliefs about quantum mechanics). Computer algorithms instead use pseudorandom generators to generate a sequence of bits from some given seed. The generators typically found on our computers usually work well but occasionally give incorrect results both in theory and in practice.

We can create theoretically better pseudorandom generators in two different ways, one based on the strong hardness assumptions of cryptography and the other based on worst-case complexity assumptions. I will focus on this second approach. We need to assume a bit more than **P** ≠ **NP**, roughly that **NP**-complete problems cannot be solved by smaller than expected AND-OR-NOT circuits. A long series of papers showed that, under this assumption, any problem with an efficient probabilistic algorithm also has an efficient algorithm that uses a pseudorandom generator with a very short seed, a surprising connection between hard languages and pseudo-randomness (see Impagliazzo). The seed is so short we can try all possible seeds efficiently and avoid the need for randomness altogether. Thus complexity theorists generally believe having randomness does not help in solving **NP** search problems and that **NP**-complete problems do not have efficient solutions, either with or without using truly random bits.

While randomness doesn't seem necessary for solving search problems, the unpredictability of random bits plays a critical role in cryptography and interactive proof systems and likely cannot be avoided in these scenarios.

37) **Could Quantum Computers Solve NP-Complete Problems?**

While we have randomized and nonrandomized efficient algorithms for determining whether a number is prime, these algorithms usually don't give us the factors of a composite number. Much of modern cryptography relies on the fact that factoring or similar problems do not have efficient algorithms. In the mid-1990s, Peter Shor showed how to factor numbers using a hypothetical quantum computer. He also developed a similar quantum algorithm to solve the discrete logarithm problem. The hardness of discrete logarithm on classical computers is also used as a basis for many cryptographic protocols. Nevertheless, we don't expect that factoring or finding discrete logarithms is **NP**-complete. While we don't think we have efficient algorithms to solve factoring or discrete logarithm, we also don't believe we can reduce **NP**-complete problems like Clique to the factoring or discrete logarithm problems.

So could quantum computers one day solve **NP**-complete problems? Unlikely.
Even if we could build these machines, **Shor's algorithm relies heavily on the algebraic structures of numbers that we don't see in the known NP-complete problems. We know that his algorithm** cannot be applied to generic "black-box" search problems so any algorithm would have to use some special structure of **NP**-complete problems that we don't know about. We have used some algebraic structure of **NP**-complete problems for interactive and zero-knowledge proofs but quantum algorithms would seem to require much more. Lov Grover did find a quantum algorithm that works on general **NP** problems but that algorithm only achieves a quadratic speed-up and we have evidence that those techniques will not go further. Meanwhile quantum cryptography, using quantum mechanics to achieve some cryptographic protocols without hardness assumptions, has had some success both in theory and in practice.

38) **A New Hope?**

Ketan Mulmuley and Milind Sohoni have presented an approach to the **P** versus **NP** problem through algebraic geometry, dubbed Geometric Complexity Theory, or GCT. This approach seems to avoid the difficulties mentioned earlier, but requires deep mathematics that could require many years or decades to carry through.
In essence, they define a family of high-dimension polygons *Pn* based on group representations on certain algebraic varieties. Roughly speaking, for each *n*, if *Pn* contains an integral point, then any circuit family for the Hamiltonian path problem must have size at least $n^{\log n}$ on inputs of size *n*, which implies **P** ≠ **NP**. Thus, to show that **P** ≠ **NP** it suffices to show that *Pn* contains an integral point for all *n*.
Although all that is necessary is to show that *Pn* contains an integral point for all *n*, Mulmuley and Sohoni argue that this direct approach would be difficult and instead suggest first showing that the integer programming problem for the family *Pn* is, in fact, in **P**. Under this approach, there are three significant steps remaining:
1. Prove that the LP relaxation solves the integer programming problem for *Pn* in polynomial time;
2. Find an efficient, simple combinatorial algorithm for the integer programming problem for *Pn*, and;

3. Prove that this simple algorithm always answers "yes."

Since the polygons *Pn* are algebro-geometric in nature, solving (1) is thought to require algebraic geometry, representation theory, and the theory of quantum groups. Mulmuley and Sohoni have given reasonable algebro-geometric conditions that imply (1). These conditions have classical analogues that are known to hold, based on the Riemann Hypothesis over finite fields (a theorem proved by André Weil in the 1960s). Mulmuley and Sohoni suggest that an analogous Riemann Hypothesis-like statement is required here (though not the classical Riemann Hypothesis).
Although step (1) is difficult, Mulmuley and Sohoni have provided definite conjectures based on reasonable mathematical analogies that would solve (1). In contrast, the path to completing steps (2) and (3) is less clear. Despite these remaining hurdles, even solving the conjectures involved in (1) could provide some insight to the **P** versus **NP** problem.
Mulmuley and Sohoni have reduced a question about the nonexistence of polynomial-time algorithms for all**NP**-complete problems to a question about the existence of a polynomial-time algorithm (with certain properties) for a specific problem. This should give us some hope, even in the face of problems (1)–(3).

Nevertheless, Mulmuley believes it will take about 100 years to carry out this program, if it works at all.

The survey focused on the **P** versus **NP** problem, its importance, our attempts to prove **P** ≠ **NP** and the approaches we use to deal with the **NP**-complete problems that nature and society throws at us. Much of the work mentioned required a long series of mathematically difficult research papers that I could not hope to adequately cover in this short article. Also the field of computational complexity goes well beyond just the **P**versus **NP** problem that I haven't discussed here. In "Further Reading," a number of references are presented for those interested in a deeper understanding of the **P** versus **NP** problem and computational complexity.
The **P** versus **NP** problem has gone from an interesting problem related to logic to perhaps the most fundamental and important mathematical question of our time, whose importance only grows as computers become more powerful and widespread. The question has even hit popular culture appearing in television shows such as *The Simpsons* and *Numb3rs*. Yet many only know of the basic principles of **P** versus **NP** and I hope this survey has given you a small feeling of the depth of research inspired by this mathematical problem.
Proving **P** ≠ **NP** would not be the end of the story, it would just show that **NP**-complete problem, don't have efficient algorithms for all inputs but many questions might remain. Cryptography, for example, would require that a problem like factoring (not believed to be **NP**-complete) is hard for randomly drawn composite numbers.
Proving **P** ≠ **NP** might not be the start of the story either. Weaker separations remain perplexingly difficult, for example showing that Boolean-formula Satisfiability cannot be solved in near-linear time or showing that some problem using a certain amount of memory cannot be solved using roughly the same amount of time.
None of us truly understands the **P** versus **NP** problem; we have only begun to peel the layers around this increasingly complex question. Perhaps we will see a resolution of the **P** versus **NP** problem in the near future but I almost hope not. The **P** versus **NP** problem continues to inspire and boggle the mind and continued exploration of this problem will lead us to yet even new complexities in that truly mysterious process we call computation.

39) A LAY MAN'S FIGMENT OF IMAGINATION:

In the 1995 Halloween episode of *The Simpsons*, Homer Simpson finds a portal to the mysterious Third Dimension behind a bookcase, and desperate to escape his in-laws, he plunges through. He finds himself wandering across a dark surface etched with green gridlines and strewn with geometric shapes, above which hover strange equations. One of these is the deceptively simple assertion that P = NP.
In fact, in a 2002 poll, 61 mathematicians and computer scientists said that they thought P probably didn't equal NP, to only nine who thought it did — and of those nine, several told the pollster that they took the position just to be contrary. But so far, no one's been able to decisively answer the question one way or the other. Frequently called the most important outstanding question in theoretical computer science, the equivalency of P and NP is one of the seven problems that the Clay Mathematics Institute will give you a million dollars for proving — or disproving. Roughly speaking, P is a set of relatively easy problems, and NP is a set of what seem to be very, very hard problems, so P = NP would imply that the apparently hard problems actually have relatively easy solutions. But the details are more complicated.

Computer science is largely concerned with a single question: How long does it take to execute a given algorithm? But computer scientists don't give the answer in minutes or milliseconds; they give it relative to the number of elements the algorithm has to manipulate.

Imagine, for instance, that you have an unsorted list of numbers, and you want to write an algorithm to find the largest one. The algorithm has to look at all the numbers in the list: there's no way around that. But if it simply keeps a record of the largest number it's seen so far, it has to look at each entry only once. The algorithm's execution time is thus directly proportional to the number of elements it's handling — which computer scientists designate N. Of course, most algorithms are more complicated, and thus less efficient, than the one for finding the largest number in a list; but many common algorithms have execution times proportional to N^2, or N times the logarithm of N, or the like.

A mathematical expression that involves N's and N^2s and N's raised to other powers is called a polynomial, and that's what the "P" in "P = NP" stands for. P is the set of problems whose solution times are proportional to polynomials involving N's.

Obviously, an algorithm whose execution time is proportional to N^3 is slower than one whose execution time is proportional to N. But such differences dwindle to insignificance compared to another distinction, between polynomial expressions — where N is the number being raised to a power — and expressions where a number is raised to the Nth power, like, say, 2^N.

If an algorithm whose execution time is proportional to N takes a second to perform a computation involving 100 elements, an algorithm whose execution time is proportional to N^3 takes almost three hours. But an algorithm whose execution time is proportional to 2^Ntakes 300 quintillion years. And that discrepancy gets much, much worse the larger N grows.

NP (which stands for nondeterministic polynomial time) is the set of problems whose solutions can be verified in polynomial time. But as far as anyone can tell, many of those problems take exponential time to solve. Perhaps the most famous problem in NP, for example, is finding prime factors of a large number. Verifying a solution just requires multiplication, but solving the problem seems to require systematically trying out lots of candidates.

So the question "Does P equal NP?" means "If the solution to a problem can be verified in polynomial time, can it be found in polynomial time?" Part of the question's allure is that the vast majority of NP problems whose solutions seem to require exponential time are what's called NP-complete, meaning that a polynomial-time solution to one can be adapted to solve all the others. And in real life, NP-complete problems are fairly common, especially in large scheduling tasks. The most famous NP-complete problem, for instance, is the so-called traveling-salesman problem: given N cities and the distances between them, can you find a route that hits all of them but is shorter than … whatever limit you choose to set?

Given that P probably doesn't equal NP, however — that efficient solutions to NP problems will probably never be found — what's all the fuss about? Michael Sipser, the head of the MIT Department of Mathematics and a member of the Computer Science and Artificial Intelligence Lab's Theory of Computation Group (TOC), says that the P-versus-NP problem is important for deepening our understanding of computational complexity.

"A major application is in the cryptography area," Sipser says, where the security of cryptographic codes is often ensured by the complexity of a computational task. The RSA cryptographic scheme, which is commonly used for secure Internet transactions — and was invented at MIT — "is really an outgrowth of the study of the complexity of doing certain number-theoretic computations," Sipser says.

Similarly, Sipser says, "the excitement around quantum computation really boiled over when Peter Shor" — another TOC member — "discovered a method for factoring numbers on a quantum computer. Peter's breakthrough inspired an enormous amount of research both in the computer science community and in the physics community." Indeed, for a while, Shor's discovery sparked the hope that quantum computers, which exploit the counterintuitive properties of extremely small particles of matter, could solve NP-complete problems in polynomial time. But that now seems unlikely: the factoring problem is actually one of the few hard NP problems that is not known to be NP-complete.

Sipser also says that "the P-versus-NP problem has become broadly recognized in the mathematical community as a mathematical question that is fundamental and important and beautiful. I think it has helped bridge the mathematics and computer science communities."

But if, as Sipser says, "complexity adds a new wrinkle on old problems" in mathematics, it's changed the questions that computer science asks. "When you're faced with a new computational problem," Sipser says, "what the theory of NP-completeness offers you is, instead of spending all of your time looking for a fast algorithm, you can spend half your time looking for a fast algorithm and the other half of your time looking for a proof of NP-completeness."

Sipser points out that some algorithms for NP-complete problems exhibit exponential complexity only in the worst-case scenario and that, in the average case, they can be more efficient than polynomial-time algorithms. But even there, NP-completeness "tells you something very specific," Sipser says. "It tells you that if you're going to look for an algorithm that's going to work in every case and give you the best solution, you're doomed: don't even try. That's useful information

40) NP-complete Problems and Physical Reality

Can NP-complete problems be solved efficiently in the physical universe? Some examples are proposals including soap bubbles, protein folding, quantum computing, quantum advice, quantum adiabatic algorithms, quantum-mechanical nonlinearities, hidden variables, relativistic time dilation, analog computing, Malament-Hogarth spacetimes, quantum gravity, closed timelike curves, and "anthropic computing." The section on soap bubbles even includes some "experimental" results

..

41) P versus NP and Exponential Time

P versus NP is the name of a question that many mathematicians, scientists, and computer programmers want to answer. P and NP are two groups of mathematical problems. P problems are considered "easy" for computers to solve. NP problems are easy only for a computer to check. For example, if you have an NP problem, and someone says "The answer to your problem is 12345," a computer can quickly figure out if the answer is right or wrong, but it may take a very long time for the computer to come up with "12345" on its own.

All P problems are NP problems, because it is easy to check that a solution is correct by solving the problem and comparing the two solutions. However, people want to know about the opposite: Are there any NP problems that are not P problems, or are all NP problems just P problems? If the NP problems are really separate from the P problems, it would mean that no fast and easy ways to solve those NP problems can exist, no matter how hard we look. However if all NP problems are P problems, it would mean that new, very fast problem solving methods do exist, but we haven't found them yet.

Since the best efforts of scientists and mathematicians haven't found the easy methods for solving NP problems yet, many people believe that there are NP problems that are not P problems. Most mathematicians also believe this to be true, but currently no one has proven it by rigorous mathematical analysis. If somehow it is proven that NP and P are the same, it would have a huge impact on many aspects of our life. For this reason the question of P versus NP has become an important and widely studied topic.

Example

Suppose a woman wants to build two towers, by stacking rocks of different mass. She wants to make sure that each of the towers has exactly the same mass. That means she will have to divide the rocks into two piles that have the same mass. If she guesses a division of the rocks that she thinks will work, it would be easy for her to check if she was right. To check her answer, she can divide the rocks into the two piles that she guessed, and then use a scale to see if they have the same mass. Because it is easy to check an answer to see if it is correct, this problem, called 'Partition' by computer scientists, is an NP problem.

If she has just 100 rocks, there are $2^{100} = 1,267,650,600,228,229,401,496,703,205,376$ possible ways to divide these rocks into two piles. For comparison, physicists believe that the universe is about $450,000,000,000,000,000$ seconds old. That means that if the woman took all of the time that has passed since the beginning of the universe, she would need to check more than $2,000,000,000,000$ different ways of dividing the rocks every second, in order to check all of those different ways.

If the woman programmed a powerful computer to test all of these ways to divide the rocks, it might be able to check $1,000,000$ ways per second. This means she would still need $2,000,000$ very powerful computers, working since the beginning of time, to test all the ways of dividing the rocks. However, perhaps it is possible for her to find a method of dividing the rocks into two equal piles without checking all combinations. The question "Is P equal to NP?" asks if any method like that exists.

Why it matters

This question is so important that the Clay Mathematical Institute will give $1,000,000 to anyone who answers it. There are many important NP problems that people don't know how to solve in a way that is faster than testing every possible answer. Here are some examples.

- A travelling salesman wants to visit 100 different cities by driving, starting and ending his trip at home. He has a limited supply of gasoline, so he can only drive a total of 10,000kilometers. He wants to know if he can visit all of the cities without running out of gasoline.

- A school offers 100 different classes, and a teacher needs to choose one hour for each class' final exam. To prevent cheating, all of the students who take a class must take the exam for that class at the same time. If a student takes more than one class, then all of those exams must be at a different time. The teacher wants to know if he can schedule all of the exams in the same day so that every student is able to take the exam for each of their classes.

- A farmer wants to take 100 watermelons of different masses to the market. She needs to pack the watermelons into boxes. Each box can only hold 20 kilograms without breaking. The farmer needs to know if 10 boxes will be enough for her to carry all 100 watermelons to market.

- A large art gallery has many rooms, and each wall is covered with many expensive paintings. The owner of the gallery wants to buy cameras to watch these paintings, in case athief tries to steal any of them. He wants to know if 100 cameras will be enough for him to make sure that each painting can be seen by at least one camera.

- The principal of a school has a list of which students are friends with each other. She wants to find the largest group of students that are all friends with each other.

Exponential Time

In the example above, we see that with 100 rocks, there are 2^{100} ways to partition the set of rocks. With n rocks, there are 2^n combinations. The function $f(n) = 2^n$ is an exponential function. It's important to NP because it models the worst-case number of computations that are needed to solve a problem and, thus, the worst-case amount of time required.

And so far, for the hard problems, the solutions have required on the order of 2^n computations. For any particular problem, people have found ways to reduce the number of computations needed. One might figure out that a way to do just 1% of the worst-case number of computation and that saves a *lot* of computing, but that is still $0.01 * (2^n)$computations. And every extra rock still doubles the number of computations needed to solve the problem. There are insights that can produce methods to do even fewer computations producing variations of the model: e.g. $2^n/n^3$. But the exponential function still dominates as n grows.

Consider the problem of scheduling exams (described above). But suppose, next, that there are 15000 students. There's a computer program that takes the schedules of all 15000 students. It runs in an hour and outputs an exam schedule so that all students can do their exams in one week. It satisfies lots of rules (no back-to-back exams, no more than 2 exams in any 28 hour period, ...) to limit the stress of exam week. The program runs for one hour at mid-term break and everyone knows his/her exam schedule with plenty of time to prepare.

The next year, though, there are 10 more students. If the same program runs on the same computer then that one hour is going to turn into 2^{10} hours, because every additional student doubles the computations. That's 6 weeks! If there were 20 more students, then

2^{20} Hours = 1048576 hours ~ 43691 days ~ 113 years

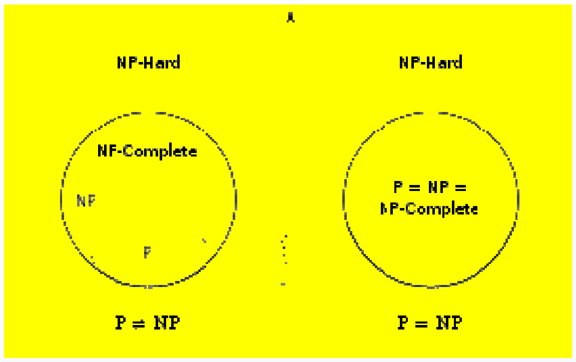
Thus, for 15000 students, it takes one hour. For 15020 students, it takes 113 years.

NP-complete problems

Mathematicians can show that there are some NP problems that are **NP-Complete**. An NP-Complete problem is at least as difficult to solve as any other NP problem. This means that if someone found a method to solve any NP-Complete problem quickly, they could use that same method to solve *every* NP problem quickly. All of the problems listed above are NP-Complete, so if the salesman found a way to plan his trip quickly, he could tell the teacher, and she could use that same method to schedule the exams. The farmer could use the same method to determine how many boxes she needs, and the woman could use the same method to find a way to build her towers.

Because a method that quickly solves one of these problems can solve them all, there are many people who want to find one. However, because there are so many different NP-Complete problems and nobody so far has found a way to solve even one of them quickly, most experts believe that is not possible to find the answers to any of them quickly

42)NP-complete problem formulation:



Euler diagram for P, NP, NP-complete, and NP-hard set of problems

In computational complexity theory, the complexity class **NP-complete** (abbreviated **NP-C** or **NPC**) is a class of decision problems. A decision problem *L* is NP-complete if it is in the set of NP problems so that any given solution to the decision problem can be verified in polynomial time, and also in the set of NP-hard problems so that any NP problem can be converted into *L* by a transformation of the inputs in polynomial time.

Although any given solution to such a problem can be verified quickly, there is no known efficient way to locate a solution in the first place; indeed, the most notable characteristic of NP-complete problems is that no fast solution to them is known. That is, the time required to solve the problem using any currently known algorithm increases very quickly as the size of the problem grows. As a result, the time required to solve even moderately sized versions of many of these problems easily reaches into the billions or trillions of years, using any amount of computing power available today. As a consequence, determining whether or not it is possible to solve these problems quickly, called the P versus NP problem, is one of the principal unsolved problems in computer science today.

While a method for computing the solutions to NP-complete problems using a reasonable amount of time remains undiscovered, computer scientists and programmers still frequently encounter NP-complete problems. NP-complete problems are often addressed by using approximation algorithms.

Formal overview

NP-complete is a subset of NP, the set of all decision problems whose solutions can be verified in polynomial time; *NP* may be equivalently defined as the set of decision problems that can be solved in polynomial time on a nondeterministic Turing machine. A problem *p* in NP is also in NPC if and only if every other problem in NP can be transformed into *p* in polynomial time. NP-complete can also be used as an adjective: problems in the class NP-complete are known as NP-complete problems.

NP-complete problems are studied because the ability to quickly verify solutions to a problem (NP) seems to correlate with the ability to quickly solve that problem (P). It is not known whether every problem in NP can be quickly solved—this is called the P = NP problem. But if *any single problem* in NP-complete can be solved quickly, then *every problem in NP* can also be quickly solved, because the definition of an NP-complete problem states that every problem in NP must be quickly reducible to every problem in NP-complete (that is, it can be reduced in polynomial time). Because of this, it is often said that the NP-complete problems are *harder* or *more difficult* than NP problems in general.

Formal definition of NP-completeness

A decision problem C is NP-complete if:

1. C is in NP, and
2. Every problem in NP is reducible to C in polynomial time.

C can be shown to be in NP by demonstrating that a candidate solution to C can be verified in polynomial time.

Note that a problem satisfying condition 2 is said to be NP-hard, whether or not it satisfies condition 1.

A consequence of this definition is that if we had a polynomial time algorithm (on a UTM, or any other Turing-equivalent abstract machine) for C, we could solve all problems in NP in polynomial time.

Background

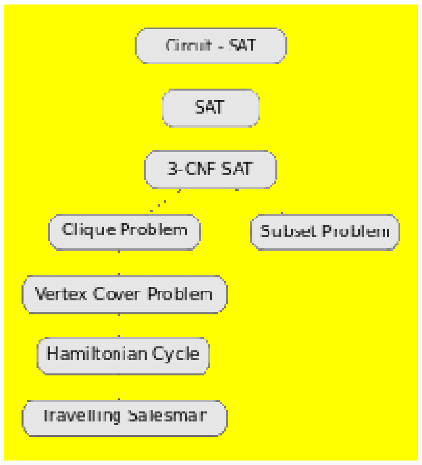
The concept of *NP-complete* was introduced in 1971 by Stephen Cook in a paper entitled *The complexity of theorem-proving procedures* on pages 151–158 of the *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, though the term *NP-complete* did not appear anywhere in his paper. At that computer science conference, there was a fierce debate among the computer scientists about whether NP-complete problems could be solved in polynomial time on a deterministic Turing machine. John Hopcroft brought everyone at the conference to a consensus that the question of whether NP-complete problems are solvable in polynomial time should be put off to be solved at some later date, since nobody had any formal proofs for their claims one way or the other. This is known as the question of whether P=NP.

Nobody has yet been able to determine conclusively whether NP-complete problems are in fact solvable in polynomial time, making this one of the great unsolved problems of mathematics. The Clay Mathematics Institute is offering a US$1 million reward to anyone who has a formal proof that P=NP or that P≠NP.

In the celebrated Cook-Levin theorem (independently proved by Leonid Levin), Cook proved that the Boolean satisfiability problem is NP-complete (a simpler, but still highly technical proof is available). In 1972, Richard Karp proved that several other problems were also NP-complete (see Karp's 21 NP-complete problems); thus there is a class of NP-complete problems (besides the Boolean satisfiability problem). Since Cook's original results, thousands of other problems have been shown to be NP-complete by reductions from other problems previously shown to be NP-complete; many of these problems are collected in Garey and Johnson's 1979 book *Computers and Intractability: A Guide to the Theory of NP-Completeness* For more details refer design and analysis of algorithm by Anany levintin.

NP-complete problems

List of NP-complete problems



Some NP-complete problems, indicating the reductions typically used to prove their NP-completeness

An interesting example is the graph isomorphism problem, the graph theory problem of determining whether a graph isomorphism exists between two graphs. Two graphs are isomorphic if one can be transformed into the other simply by renaming vertices. Consider these two problems:

- Graph Isomorphism: Is graph G_1 isomorphic to graph G_2?
- Sub graph Isomorphism: Is graph G_1 isomorphic to a sub graph of graph G_2?

The Sub graph Isomorphism problem is NP-complete. The graph isomorphism problem is suspected to be neither in P nor NP-complete, though it is in NP. This is an example of a problem that is thought to be **hard**, but isn't thought to be NP-complete.

The easiest way to prove that some new problem is NP-complete is first to prove that it is in NP, and then to reduce some known NP-complete problem to it. Therefore, it is useful to know a variety of NP-complete problems. The list below contains some well-known problems that are NP-complete when expressed as decision problems.

- Boolean satisfiability problem (Sat.)
- N-puzzle
- Knapsack problem
- Hamiltonian path problem
- Travelling salesman problem
- Sub graph isomorphism problem
- Subset sum problem
- Clique problem
- Vertex cover problem
- Independent set problem
- Dominating set problem
- Graph coloring problem

To the right is a diagram of some of the problems and the reductions typically used to prove their NP-completeness. In this diagram, an arrow from one problem to another indicates the direction of the reduction. Note that this diagram is misleading as a description of the mathematical relationship between these problems, as there exists a polynomial-time reduction between any two NP-complete problems; but it indicates where demonstrating this polynomial-time reduction has been easiest.

There is often only a small difference between a problem in P and an NP-complete problem. For example, the 3-satisfiability problem, a restriction of the boolean satisfiability problem, remains NP-complete, whereas the slightly more restricted 2-satisfiability problem is in P (specifically, NL-complete), and the slightly more general max. 2-sat. Problem is again NP-complete. Determining whether a graph can be colored with 2 colors is in P, but with 3 colors is NP-complete, even when restricted to planar graphs.

Determining if a graph is a cycle or is bipartite is very easy (in L), but finding a maximum bipartite or a maximum cycle sub graph is NP-complete. A solution of the knapsack problem within any fixed percentage of the optimal solution can be computed in polynomial time, but finding the optimal solution is NP-complete.

Solving NP-complete problems

At present, all known algorithms for NP-complete problems require time that is super polynomial in the input size, and it is unknown whether there are any faster algorithms.

The following techniques can be applied to solve computational problems in general, and they often give rise to substantially faster algorithms:

- Approximation: Instead of searching for an optimal solution, search for an "almost" optimal one.
- Randomization: Use randomness to get a faster average running time, and allow the algorithm to fail with some small probability. Note: The Monte Carlo method is not an example of an efficient algorithm, although evolutionary approaches like Genetic algorithms may be.
- Restriction: By restricting the structure of the input (e.g., to planar graphs), faster algorithms are usually possible.
- Parameterization: Often there are fast algorithms if certain parameters of the input are fixed.
- Heuristic: An algorithm that works "reasonably well" in many cases, but for which there is no proof that it is both always fast and always produces a good result. Metaheuristicapproaches are often used.

One example of a heuristic algorithm is a suboptimal $O(n \log n)$ greedy coloring algorithm used for graph coloring during the register allocation phase of some compilers, a technique called graph-coloring global register allocation. Each vertex is a variable, edges are drawn between variables which are being used at the same time, and colors indicate the register assigned to each variable. Because most RISC machines have a fairly large number of general-purpose registers, even a heuristic approach is effective for this application.

Completeness under different types of reduction

In the definition of NP-complete given above, the term *reduction* was used in the technical meaning of a polynomial-time many-one reduction.

Another type of reduction is polynomial-time Turing reduction. A problem X is polynomial-time Turing-reducible to a problem Y if, given a subroutine that solves Y in polynomial time, one could write a program that calls this subroutine and solves X in polynomial time. This contrasts with many-one reducibility, which has the restriction that the program can only call the subroutine once, and the return value of the subroutine must be the return value of the program.

If one defines the analogue to NP-complete with Turing reductions instead of many-one reductions, the resulting set of problems won't be smaller than NP-complete; it is an open question whether it will be any larger.

Another type of reduction that is also often used to define NP-completeness is the logarithmic-space many-one reduction which is a many-one reduction that can be computed with only a logarithmic amount of space. Since every computation that can be done in logarithmic space can also be done in polynomial time it follows that if there is a logarithmic-space many-one reduction then there is also a polynomial-time many-one reduction. This type of reduction is more refined than the more usual polynomial-time many-one reductions and it allows us to distinguish more classes such as P-complete. Whether under these types of reductions the definition of NP-complete changes is still an open problem. All currently known NP-complete problems are NP-complete under log space reductions. Indeed, all currently known NP-complete problems remain NP-complete even under much weaker reductions.[2] It is known, however, that AC^0 reductions define a strictly smaller class than polynomial-time reductions.

Naming

According to Don Knuth, the name "NP-complete" was popularized by Alfred Aho, John Hopcroft and Jeffrey Ullman in their celebrated textbook "The Design and Analysis of Computer Algorithms". He reports that they introduced the change in the galley proofs for the book (from "polynomially-complete"), in accordance with the results of a poll he had conducted of the Theoretical Computer Science community. Other suggestions made in the poll included "Herculean", "formidable", Steiglitz's "hard-boiled" in honor of Cook, and Shen Lin's acronym "PET", which stood for "probably exponential time", but depending on which way the P versus NP problem went, could stand for "provably exponential time" or "previously exponential time"

Common misconceptions

The following misconceptions are frequent

- *"NP-complete problems are the most difficult known problems."* Since NP-complete problems are in NP, their running time is at most exponential. However, some problems provably require more time, for example Presburger arithmetic.
- *"NP-complete problems are difficult because there are so many different solutions."* On the one hand, there are many problems that have a solution space just as large, but can be solved in polynomial time (for example minimum spanning tree). On the other hand, there are NP-problems with at most one solution that is NP-hard under randomized polynomial-time reduction (see Valiant–Vazirani theorem).
- *"Solving NP-complete problems requires exponential time."* First, this would imply P ≠ NP, which is still an unsolved question. Further, some NP-complete problems actually have algorithms running in super polynomial, but sub exponential time. For example, the Independent set and dominating set problems are NP-complete when restricted to planar graphs, but can be solved in sub exponential time on planar graphs using the planar separator theorem.
- *"All instances of an NP-complete problem are difficult."* Often some instances, or even almost all instances, may be easy to solve within polynomial time.

P VIS A VIS NP SYSTEM CONCATENATED WITH CONCOMITANT "NPC"-NP(HARD) SYSTEM: EXPOSITION OF FUNDAMENTAL VARIABLES:

NOTATION :

G_{13} : CATEGORY ONE OF P VIS A VIS NP SYSTEM CONCATENATED WITH CONCOMITANT "NPC"-NP(HARD) SYSTEM

G_{14} : CATEGORY TWO OF P VIS A VIS NP SYSTEM CONCATENATED WITH CONCOMITANT "NPC"-NP(HARD) SYSTEM

G_{15} : CATEGORY THREE OF P VIS A VIS NP SYSTEM CONCATENATED WITH CONCOMITANT "NPC"-NP(HARD) SYSTEM

T_{13} : CATEGORY ONE OF "NP" VIS A VIS "P" CONSUMMATED WITH CORRESPONDING SYSTEM OF "(NPC) AND "NP(HARD)"

T_{14} : CATEGORY TWO OF P VIS A VIS NP SYSTEM CONCATENATED WITH CONCOMITANT "NPC"-NP(HARD) SYSTEM

T_{15} : CATEGORY THREE OF P VIS A VIS NP SYSTEM CONCATENATED WITH CONCOMITANT "NPC"-NP(HARD) SYSTEM

G_{16} : CATEGORY ONE OF "NPC" VIS A VIS "NP(HARD)" CONSUNSTATNTIATED IN THE ANTERIOR TO THE SYATEM "P" VIS A VIS "NP"

G_{17} : CATEGORY TWO OF P VIS A VIS NP SYSTEM CONCATENATED WITH CONCOMITANT "NPC"-NP(HARD) SYSTEM

G_{18} : CATEGORY THREE OF P VIS A VIS NP SYSTEM CONCATENATED WITH CONCOMITANT "NPC"-NP(HARD) SYSTEM

T_{16} : CATEGORY ONE OF NP(HARD)" SYSTEM VIS A VIS "NPC" CONCRETISED AND

FORTIFIED WITH THE SYSTEM "P" VIS V VIS "NP"

T_{17} : CATEGORY TWO OF NP(HARD)" SYSTEM VIS A VIS "NPC" CONCRETISED AND FORTIFIED WITH THE SYSTEM "P" VIS V VIS "NP

T_{18} : CATEGORY THREE OF NP(HARD" SYSTEM VIS A VIS "NPC" CONCRETISED AND FORTIFIED WITH THE SYSTEM "P" VIS V VIS "NP

$(a_{13})^{(1)}, (a_{14})^{(1)}, (a_{15})^{(1)}$, $(b_{13})^{(1)}, (b_{14})^{(1)}, (b_{15})^{(1)}$ $(a_{16})^{(2)}, (a_{17})^{(2)}, (a_{18})^{(2)}$ $(b_{16})^{(2)}, (b_{17})^{(2)}, (b_{18})^{(2)}$: are Accentuation coefficients

$(a'_{13})^{(1)}, (a'_{14})^{(1)}, (a'_{15})^{(1)}$, $(b'_{13})^{(1)}, (b'_{14})^{(1)}, (b'_{15})^{(1)}$, $(a'_{16})^{(2)}, (a'_{17})^{(2)}, (a'_{18})^{(2)}$, $(b'_{16})^{(2)}, (b'_{17})^{(2)}, (b'_{18})^{(2)}$ are Dissipation coefficients

<u>"P" AND "NP" SYSTEM:GOVERNING EQUATIONS</u>:

The differential system of this model is now

$$\frac{dG_{13}}{dt} = (a_{13})^{(1)}G_{14} - \left[(a'_{13})^{(1)} + (a''_{13})^{(1)}(T_{14},t)\right]G_{13} \quad 1$$

$$\frac{dG_{14}}{dt} = (a_{14})^{(1)}G_{13} - \left[(a'_{14})^{(1)} + (a''_{14})^{(1)}(T_{14},t)\right]G_{14} \quad 2$$

$$\frac{dG_{15}}{dt} = (a_{15})^{(1)}G_{14} - \left[(a'_{15})^{(1)} + (a''_{15})^{(1)}(T_{14},t)\right]G_{15} \quad 3$$

$$\frac{dT_{13}}{dt} = (b_{13})^{(1)}T_{14} - \left[(b'_{13})^{(1)} - (b''_{13})^{(1)}(G,t)\right]T_{13} \quad 4$$

$$\frac{dT_{14}}{dt} = (b_{14})^{(1)}T_{13} - \left[(b'_{14})^{(1)} - (b''_{14})^{(1)}(G,t)\right]T_{14} \quad 5$$

$$\frac{dT_{15}}{dt} = (b_{15})^{(1)}T_{14} - \left[(b'_{15})^{(1)} - (b''_{15})^{(1)}(G,t)\right]T_{15} \quad 6$$

$+(a''_{13})^{(1)}(T_{14},t) =$ First augmentation factor 7

$-(b''_{13})^{(1)}(G,t) =$ First detritions factor 8

<u>NPC VIS A VIS NP(HARD) SYSTEM -GOVERNING EQUATIONS</u>:

The differential system of this model is now

$$\frac{dG_{16}}{dt} = (a_{16})^{(2)}G_{17} - \left[(a'_{16})^{(2)} + (a''_{16})^{(2)}(T_{17},t)\right]G_{16} \quad 9$$

$$\frac{dG_{17}}{dt} = (a_{17})^{(2)}G_{16} - \left[(a'_{17})^{(2)} + (a''_{17})^{(2)}(T_{17},t)\right]G_{17} \quad 10$$

$$\frac{dG_{18}}{dt} = (a_{18})^{(2)}G_{17} - \left[(a'_{18})^{(2)} + (a''_{18})^{(2)}(T_{17},t)\right]G_{18} \quad 11$$

$$\frac{dT_{16}}{dt} = (b_{16})^{(2)}T_{17} - \left[(b'_{16})^{(2)} - (b''_{16})^{(2)}\big((G_{19}),t\big)\right]T_{16} \quad 12$$

$$\frac{dT_{17}}{dt} = (b_{17})^{(2)}T_{16} - \left[(b'_{17})^{(2)} - (b''_{17})^{(2)}\big((G_{19}),t\big)\right]T_{17} \quad 13$$

$$\frac{dT_{18}}{dt} = (b_{18})^{(2)}T_{17} - \left[(b'_{18})^{(2)} - (b''_{18})^{(2)}\big((G_{19}),t\big)\right]T_{18} \quad 14$$

$+(a''_{16})^{(2)}(T_{17},t) =$ First augmentation factor 15

$-(b''_{16})^{(2)}\big((G_{19}),t\big) =$ First detritions factor 16

<u>GOVERNING EQUATIONS OF THE SYTEM "P VIS A VIS "NP" CONCATENATED WITH</u>

CORRESPONDING AND CONCOMITANT "NPC" VIS A VIS NP(HARD)

$$\frac{dG_{13}}{dt} = (a_{13})^{(1)}G_{14} - \left[(a'_{13})^{(1)}\boxed{+(a''_{13})^{(1)}(T_{14},t)}\boxed{+(a''_{16})^{(2,2)}(T_{17},t)}\right]G_{13} \quad 17$$

$$\frac{dG_{14}}{dt} = (a_{14})^{(1)}G_{13} - \left[(a'_{14})^{(1)}\boxed{+(a''_{14})^{(1)}(T_{14},t)}\boxed{+(a''_{17})^{(2,2)}(T_{17},t)}\right]G_{14} \quad 18$$

$$\frac{dG_{15}}{dt} = (a_{15})^{(1)}G_{14} - \left[(a'_{15})^{(1)}\boxed{+(a''_{15})^{(1)}(T_{14},t)}\boxed{+(a''_{18})^{(2,2)}(T_{17},t)}\right]G_{15} \quad 19$$

Where $\boxed{(a''_{13})^{(1)}(T_{14},t)}$, $\boxed{(a''_{14})^{(1)}(T_{14},t)}$, $\boxed{(a''_{15})^{(1)}(T_{14},t)}$ are first augmentation coefficients for category 1, 2 and 3

$\boxed{+(a''_{16})^{(2,2)}(T_{17},t)}$, $\boxed{+(a''_{17})^{(2,2)}(T_{17},t)}$, $\boxed{+(a''_{18})^{(2,2)}(T_{17},t)}$ are second augmentation coefficients for category 1, 2 and 3

$$\frac{dT_{13}}{dt} = (b_{13})^{(1)}T_{14} - \left[(b'_{13})^{(1)}\boxed{-(b''_{13})^{(1)}(G,t)}\boxed{+(b''_{16})^{(2,2)}(G_{19},t)}\right]T_{13} \quad 20$$

$$\frac{dT_{14}}{dt} = (b_{14})^{(1)}T_{13} - \left[(b'_{14})^{(1)}\boxed{-(b''_{14})^{(1)}(G,t)}\boxed{+(b''_{17})^{(2,2)}(G_{19},t)}\right]T_{14} \quad 21$$

$$\frac{dT_{15}}{dt} = (b_{15})^{(1)}T_{14} - \left[(b'_{15})^{(1)}\boxed{-(b''_{15})^{(1)}(G,t)}\boxed{+(b''_{18})^{(2,2)}(G_{19},t)}\right]T_{15} \quad 22$$

Where $\boxed{-(b''_{13})^{(1)}(G,t)}$, $\boxed{-(b''_{14})^{(1)}(G,t)}$, $\boxed{-(b''_{15})^{(1)}(G,t)}$ are first detrition coefficients for category 1, 2 and 3 23

$\boxed{+(b''_{16})^{(2,2)}(G_{19},t)}$, $\boxed{+(b''_{17})^{(2,2)}(G_{19},t)}$, $\boxed{+(b''_{18})^{(2,2)}(G_{19},t)}$ are second augmentation coefficients for category 1, 2 and 3

$$\frac{dG_{16}}{dt} = (a_{16})^{(2)}G_{17} - \left[(a'_{16})^{(2)}\boxed{+(a''_{16})^{(2)}(T_{17},t)}\boxed{+(a''_{13})^{(1,1)}(T_{14},t)}\right]G_{16} \quad 24$$

$$\frac{dG_{17}}{dt} = (a_{17})^{(2)}G_{16} - \left[(a'_{17})^{(2)}\boxed{+(a''_{17})^{(2)}(T_{17},t)}\boxed{+(a''_{14})^{(1,1)}(T_{14},t)}\right]G_{17} \quad 25$$

$$\frac{dG_{18}}{dt} = (a_{18})^{(2)}G_{17} - \left[(a'_{18})^{(2)}\boxed{+(a''_{18})^{(2)}(T_{17},t)}\boxed{+(a''_{15})^{(1,1)}(T_{14},t)}\right]G_{18} \quad 26$$

Where $\boxed{+(a''_{16})^{(2)}(T_{17},t)}$, $\boxed{+(a''_{17})^{(2)}(T_{17},t)}$, $\boxed{+(a''_{18})^{(2)}(T_{17},t)}$ are first augmentation coefficients for category 1, 2 and 3

$\boxed{+(a''_{13})^{(1,1)}(T_{14},t)}$, $\boxed{+(a''_{14})^{(1,1)}(T_{14},t)}$, $\boxed{+(a''_{15})^{(1,1)}(T_{14},t)}$ are second detrition coefficients for category 1, 2 and 3

$$\frac{dT_{16}}{dt} = (b_{16})^{(2)}T_{17} - \left[(b'_{16})^{(2)}\boxed{-(b''_{16})^{(2)}(G_{19},t)}\boxed{-(b''_{13})^{(1,1)}(G,t)}\right]T_{16} \quad 27$$

$$\frac{dT_{17}}{dt} = (b_{17})^{(2)}T_{16} - \left[(b'_{17})^{(2)}\boxed{-(b''_{17})^{(2)}(G_{19},t)}\boxed{-(b''_{14})^{(1,1)}(G,t)}\right]T_{17} \quad 28$$

$$\frac{dT_{18}}{dt} = (b_{18})^{(2)}T_{17} - \left[(b'_{18})^{(2)}\boxed{-(b''_{18})^{(2)}(G_{19},t)}\boxed{-(b''_{15})^{(1,1)}(G,t)}\right]T_{18} \quad 29$$

Where $\boxed{-(b_{16}^{''})^{(2)}(G_{19},t)}$, $\boxed{-(b_{17}^{''})^{(2)}(G_{19},t)}$, $\boxed{-(b_{18}^{''})^{(2)}(G_{19},t)}$ are first detrition coefficients for category 1, 2 and 3

$\boxed{-(b_{13}^{''})^{(1,1)}(G,t)}$, $\boxed{-(b_{14}^{''})^{(1,1)}(G,t)}$, $\boxed{-(b_{15}^{''})^{(1,1)}(G,t)}$ are second detrition coefficients for category 1, 2 and 3

Where we suppose

(A) $(a_i)^{(1)}, (a_i^{'})^{(1)}, (a_i^{''})^{(1)}, (b_i)^{(1)}, (b_i^{'})^{(1)}, (b_i^{''})^{(1)} > 0,$ *30*

$i, j = 13,14,15$

(B) The functions $(a_i^{''})^{(1)}, (b_i^{''})^{(1)}$ are positive continuous increasing and bounded.

<u>**Definition of**</u> $(p_i)^{(1)}, \ (r_i)^{(1)}$:

$$(a_i^{''})^{(1)}(T_{14},t) \le (p_i)^{(1)} \le (\hat{A}_{13})^{(1)}$$

$$(b_i^{''})^{(1)}(G,t) \le \ (r_i)^{(1)} \le (b_i^{'})^{(1)} \le (\hat{B}_{13})^{(1)}$$

(C) $lim_{T_2 \to \infty} (a_i^{''})^{(1)} (T_{14},t) = (p_i)^{(1)}$ 31

$$\lim_{G \to \infty} (b_i^{''})^{(1)} (G,t) = \ (r_i)^{(1)}$$

<u>**Definition of**</u> $(\hat{A}_{13})^{(1)}, (\hat{B}_{13})^{(1)}$:

Where $\boxed{(\hat{A}_{13})^{(1)}, (\hat{B}_{13})^{(1)}, (p_i)^{(1)}, \ (r_i)^{(1)}}$ are positive constants and $\boxed{i = 13,14,15}$

They satisfy Lipschitz condition:

$$|(a_i^{''})^{(1)}(T_{14}^{'},t) - (a_i^{''})^{(1)}(T_{14},t)| \le (\hat{k}_{13})^{(1)}|T_{14} - T_{14}^{'}|e^{-(\hat{M}_{13})^{(1)}t}$$

$$|(b_i^{''})^{(1)}(G^{'},t) - (b_i^{''})^{(1)}(G,T)| < (\hat{k}_{13})^{(1)}||G - G^{'}||e^{-(\hat{M}_{13})^{(1)}t}$$ 32

With the Lipschitz condition, we place a restriction on the behavior of functions $(a_i^{''})^{(1)}(T_{14}^{'},t)$ and$(a_i^{''})^{(1)}(T_{14},t)$. $(T_{14}^{'},t)$ and (T_{14},t) are points belonging to the interval $\left[(\hat{k}_{13})^{(1)}, (\hat{M}_{13})^{(1)}\right]$. It is to be noted that $(a_i^{''})^{(1)}(T_{14},t)$ is uniformly continuous. In the eventuality of the fact, that if $(\hat{M}_{13})^{(1)} = 1$ then the function $(a_i^{''})^{(1)}(T_{14},t)$, the first augmentation coefficient would be absolutely continuous.

<u>**Definition of**</u> $(\hat{M}_{13})^{(1)}, (\hat{k}_{13})^{(1)}$: 33

(D) $(\hat{M}_{13})^{(1)}, (\hat{k}_{13})^{(1)},$ are positive constants

$$\frac{(a_i)^{(1)}}{(\hat{M}_{13})^{(1)}} , \frac{(b_i)^{(1)}}{(\hat{M}_{13})^{(1)}} < 1$$

<u>**Definition of**</u> $(\hat{P}_{13})^{(1)}, (\hat{Q}_{13})^{(1)}$:

(E) There exists two constants $(\hat{P}_{13})^{(1)}$ and $(\hat{Q}_{13})^{(1)}$ which together with $(\hat{M}_{13})^{(1)}, (\hat{k}_{13})^{(1)}, (\hat{A}_{13})^{(1)} and \ (\hat{B}_{13})^{(1)}$ and the constants $(a_i)^{(1)}, (a_i^{'})^{(1)}, (b_i)^{(1)}, (b_i^{'})^{(1)}, (p_i)^{(1)}, \ (r_i)^{(1)}, i = 13,14,15,$

satisfy the inequalities 34

$\frac{1}{(\hat{M}_{13})^{(1)}}[\,(a_i)^{(1)}+(a_i')^{(1)}+\ (\hat{A}_{13})^{(1)}+\ (\hat{P}_{13})^{(1)}\,(\hat{k}_{13})^{(1)}]<1$

$\frac{1}{(\hat{M}_{13})^{(1)}}[\ (b_i)^{(1)}+(b_i')^{(1)}+\ (\hat{B}_{13})^{(1)}+\ (\hat{Q}_{13})^{(1)}\ (\hat{k}_{13})^{(1)}]<1$

Where we suppose

(F) $(a_i)^{(2)},(a_i')^{(2)},(a_i'')^{(2)},(b_i)^{(2)},(b_i')^{(2)},(b_i'')^{(2)}>0,\quad i,j=16,17,18$ 35

(G) The functions $(a_i'')^{(2)},(b_i'')^{(2)}$ are positive continuous increasing and bounded. 36

<u>Definition of</u> $(p_i)^{(2)},\ (r_i)^{(2)}$: 37

$(a_i'')^{(2)}(T_{17},t)\le(p_i)^{(2)}\le(\hat{A}_{16})^{(2)}$ 38

$(b_i'')^{(2)}(G,t)\le\ (r_i)^{(2)}\le(b_i')^{(2)}\le(\hat{B}_{16})^{(2)}$ 39

(H) $\lim_{T_2\to\infty}(a_i'')^{(2)}(T_{17},t)=(p_i)^{(2)}$ 40

$\lim_{G\to\infty}(b_i'')^{(2)}\big((G_{19}),t\big)=\ (r_i)^{(2)}$ 41

<u>Definition of</u> $(\hat{A}_{16})^{(2)},(\hat{B}_{16})^{(2)}$: 42

Where $\boxed{(\hat{A}_{16})^{(2)},(\hat{B}_{16})^{(2)},(p_i)^{(2)},\ (r_i)^{(2)}}$ are positive constants and $\boxed{i=16,17,18}$

They satisfy Lipschitz condition:

$|(a_i'')^{(2)}(T_{17}',t)-(a_i'')^{(2)}(T_{17},t)|\le(\hat{k}_{16})^{(2)}|T_{17}-T_{17}'|e^{-(\hat{M}_{16})^{(2)}t}$ 43

$|(b_i'')^{(2)}((G_{19})',t)-(b_i'')^{(2)}\big((G_{19}),T_{19}\big)|<(\hat{k}_{16})^{(2)}||(G_{19})-(G_{19})'||e^{-(\hat{M}_{16})^{(2)}t}$ 44

With the Lipschitz condition, we place a restriction on the behavior of functions $(a_i'')^{(2)}(T_{17}',t)$ and$(a_i'')^{(2)}(T_{17},t)$. (T_{17}',t) And (T_{17},t) are points belonging to the interval $\big[(\hat{k}_{16})^{(2)},(\hat{M}_{16})^{(2)}\big]$. It is to be noted that $(a_i'')^{(2)}(T_{17},t)$ is uniformly continuous. In the eventuality of the fact, that if $(\hat{M}_{16})^{(2)}=1$ then the function $(a_i'')^{(2)}(T_{17},t)$, the first augmentation coefficient WOULD BE absolutely continuous.

<u>Definition of</u> $(\hat{M}_{16})^{(2)},(\hat{k}_{16})^{(2)}$: 45

(I) $(\hat{M}_{16})^{(2)},(\hat{k}_{16})^{(2)},$ are positive constants 46

$\frac{(a_i)^{(2)}}{(\hat{M}_{16})^{(2)}}\ ,\frac{(b_i)^{(2)}}{(\hat{M}_{16})^{(2)}}<1$

<u>Definition of</u> $(\hat{P}_{13})^{(2)},(\hat{Q}_{13})^{(2)}$: 47

There exists two constants $(\hat{P}_{16})^{(2)}$ and $(\hat{Q}_{16})^{(2)}$ which together with $(\hat{M}_{16})^{(2)},(\hat{k}_{16})^{(2)},(\hat{A}_{16})^{(2)}and\ (\hat{B}_{16})^{(2)}$ and the constants $(a_i)^{(2)},(a_i')^{(2)},(b_i)^{(2)},(b_i')^{(2)},(p_i)^{(2)},\ (r_i)^{(2)},i=16,17,18,$

satisfy the inequalities

$\frac{1}{(\hat{M}_{16})^{(2)}}[\,(a_i)^{(2)}+(a_i')^{(2)}+\ (\hat{A}_{16})^{(2)}+\ (\hat{P}_{16})^{(2)}\,(\hat{k}_{16})^{(2)}]<1$ 48

$\frac{1}{(\hat{M}_{16})^{(2)}}[\ (b_i)^{(2)}+(b_i')^{(2)}+\ (\hat{B}_{16})^{(2)}+\ (\hat{Q}_{16})^{(2)}\ (\hat{k}_{16})^{(2)}]<1$ 49

Theorem 1: if the conditions (A)-(I) above are fulfilled, there exists a solution satisfying the conditions 50

Definition of $G_i(0)\,,T_i(0)$:

$G_i(t) \leq \left(\hat{P}_{13}\right)^{(1)} e^{(\hat{M}_{13})^{(1)}t}$, $\boxed{G_i(0) = G_i^0 > 0}$

$T_i(t) \leq (\hat{Q}_{13})^{(1)} e^{(\hat{M}_{13})^{(1)}t}$, $\boxed{T_i(0) = T_i^0 > 0}$

51

Definition of $G_i(0)\,,T_i(0)$

$G_i(t) \leq (\hat{P}_{16})^{(2)} e^{(\hat{M}_{16})^{(2)}t}$, $G_i(0) = G_i^0 > 0$

$T_i(t) \leq (\hat{Q}_{16})^{(2)} e^{(\hat{M}_{16})^{(2)}t}$, $T_i(0) = T_i^0 > 0$

PROOF: 52

Consider operator $\mathcal{A}^{(1)}$ defined on the space of sextuples of continuous functions $G_i, T_i: \mathbb{R}_+ \rightarrow \mathbb{R}_+$ which satisfy

$G_i(0) = G_i^0\,, T_i(0) = T_i^0\,, G_i^0 \leq (\hat{P}_{13})^{(1)}, T_i^0 \leq (\hat{Q}_{13})^{(1)},$ 53

$0 \leq G_i(t) - G_i^0 \leq (\hat{P}_{13})^{(1)} e^{(\hat{M}_{13})^{(1)}t}$ 54

$0 \leq T_i(t) - T_i^0 \leq (\hat{Q}_{13})^{(1)} e^{(\hat{M}_{13})^{(1)}t}$ 55

By 56

$\bar{G}_{13}(t) = G_{13}^0 + \int_0^t \left[(a_{13})^{(1)} G_{14}(s_{(13)}) - \left((a'_{13})^{(1)} + a''_{13})^{(1)}(T_{14}(s_{(13)}), s_{(13)})\right) G_{13}(s_{(13)})\right] ds_{(13)}$

$\bar{G}_{14}(t) = G_{14}^0 + \int_0^t \left[(a_{14})^{(1)} G_{13}(s_{(13)}) - \left((a'_{14})^{(1)} + (a''_{14})^{(1)}(T_{14}(s_{(13)}), s_{(13)})\right) G_{14}(s_{(13)})\right] ds_{(13)}$ 57

$\bar{G}_{15}(t) = G_{15}^0 + \int_0^t \left[(a_{15})^{(1)} G_{14}(s_{(13)}) - \left((a'_{15})^{(1)} + (a''_{15})^{(1)}(T_{14}(s_{(13)}), s_{(13)})\right) G_{15}(s_{(13)})\right] ds_{(13)}$ 58

$\bar{T}_{13}(t) = T_{13}^0 + \int_0^t \left[(b_{13})^{(1)} T_{14}(s_{(13)}) - \left((b'_{13})^{(1)} - (b''_{13})^{(1)}(G(s_{(13)}), s_{(13)})\right) T_{13}(s_{(13)})\right] ds_{(13)}$ 59

$\bar{T}_{14}(t) = T_{14}^0 + \int_0^t \left[(b_{14})^{(1)} T_{13}(s_{(13)}) - \left((b'_{14})^{(1)} - (b''_{14})^{(1)}(G(s_{(13)}), s_{(13)})\right) T_{14}(s_{(13)})\right] ds_{(13)}$ 60

$\bar{\mathrm{T}}_{15}(\mathrm{t}) = \mathrm{T}_{15}^0 + \int_0^t \left[(b_{15})^{(1)} T_{14}(s_{(13)}) - \left((b'_{15})^{(1)} - (b''_{15})^{(1)}(G(s_{(13)}), s_{(13)})\right) T_{15}(s_{(13)})\right] ds_{(13)}$ 61

Where $s_{(13)}$ is the integrand that is integrated over an interval $(0, t)$

PROOF:

Consider operator $\mathcal{A}^{(2)}$ defined on the space of sextuples of continuous functions $G_i, T_i: \mathbb{R}_+ \rightarrow \mathbb{R}_+$ which satisfy

$G_i(0) = G_i^0\,, T_i(0) = T_i^0\,, G_i^0 \leq (\hat{P}_{16})^{(2)}, T_i^0 \leq (\hat{Q}_{16})^{(2)},$ 62

$0 \leq G_i(t) - G_i^0 \leq (\hat{P}_{16})^{(2)} e^{(\hat{M}_{16})^{(2)}t}$ 63

$0 \leq T_i(t) - T_i^0 \leq (\hat{Q}_{16})^{(2)} e^{(\hat{M}_{16})^{(2)} t}$ 64

By 65

$\bar{G}_{16}(t) = G_{16}^0 + \int_0^t \left[(a_{16})^{(2)} G_{17}(s_{(16)}) - \left((a'_{16})^{(2)} + a''_{16})^{(2)} (T_{17}(s_{(16)}), s_{(16)}) \right) G_{16}(s_{(16)}) \right] ds_{(16)}$

$\bar{G}_{17}(t) = G_{17}^0 + \int_0^t \left[(a_{17})^{(2)} G_{16}(s_{(16)}) - \left((a'_{17})^{(2)} + (a''_{17})^{(2)} (T_{17}(s_{(16)}), s_{(17)}) \right) G_{17}(s_{(16)}) \right] ds_{(16)}$ 66

$\bar{G}_{18}(t) = G_{18}^0 + \int_0^t \left[(a_{18})^{(2)} G_{17}(s_{(16)}) - \left((a'_{18})^{(2)} + (a''_{18})^{(2)} (T_{17}(s_{(16)}), s_{(16)}) \right) G_{18}(s_{(16)}) \right] ds_{(16)}$ 67

$\bar{T}_{16}(t) = T_{16}^0 + \int_0^t \left[(b_{16})^{(2)} T_{17}(s_{(16)}) - \left((b'_{16})^{(2)} - (b''_{16})^{(2)} (G(s_{(16)}), s_{(16)}) \right) T_{16}(s_{(16)}) \right] ds_{(16)}$ 68

$\bar{T}_{17}(t) = T_{17}^0 + \int_0^t \left[(b_{17})^{(2)} T_{16}(s_{(16)}) - \left((b'_{17})^{(2)} - (b''_{17})^{(2)} (G(s_{(16)}), s_{(16)}) \right) T_{17}(s_{(16)}) \right] ds_{(16)}$ 69

$\bar{T}_{18}(t) = T_{18}^0 + \int_0^t \left[(b_{18})^{(2)} T_{17}(s_{(16)}) - \left((b'_{18})^{(2)} - (b''_{18})^{(2)} (G(s_{(16)}), s_{(16)}) \right) T_{18}(s_{(16)}) \right] ds_{(16)}$ 70

Where $s_{(16)}$ is the integrand that is integrated over an interval $(0, t)$

(a) The operator $\mathcal{A}^{(1)}$ maps the space of functions into itself .Indeed it is obvious that 71

$G_{13}(t) \leq G_{13}^0 + \int_0^t \left[(a_{13})^{(1)} \left(G_{14}^0 + (\hat{P}_{13})^{(1)} e^{(\hat{M}_{13})^{(1)} s_{(13)}} \right) \right] ds_{(13)} =$

$$\left(1 + (a_{13})^{(1)} t\right) G_{14}^0 + \frac{(a_{13})^{(1)} (\hat{P}_{13})^{(1)}}{(\hat{M}_{13})^{(1)}} \left(e^{(\hat{M}_{13})^{(1)} t} - 1 \right)$$

From which it follows that 72

$$(G_{13}(t) - G_{13}^0) e^{-(\hat{M}_{13})^{(1)} t} \leq \frac{(a_{13})^{(1)}}{(\hat{M}_{13})^{(1)}} \left[\left((\hat{P}_{13})^{(1)} + G_{14}^0 \right) e^{\left(-\frac{(\hat{P}_{13})^{(1)} + G_{14}^0}{G_{14}^0} \right)} + (\hat{P}_{13})^{(1)} \right]$$

(G_i^0) is as defined in the statement of theorem 1

Analogous inequalities hold also for $G_{14}, G_{15}, T_{13}, T_{14}, T_{15}$

(b) The operator $\mathcal{A}^{(2)}$ maps the space of functions into itself .Indeed it is obvious that

$G_{16}(t) \leq G_{16}^0 + \int_0^t \left[(a_{16})^{(2)} \left(G_{17}^0 + (\hat{P}_{16})^{(6)} e^{(\hat{M}_{16})^{(2)} s_{(16)}} \right) \right] ds_{(16)} =$ 73

$$\left(1 + (a_{16})^{(2)} t\right) G_{17}^0 + \frac{(a_{16})^{(2)} (\hat{P}_{16})^{(2)}}{(\hat{M}_{16})^{(2)}} \left(e^{(\hat{M}_{16})^{(2)} t} - 1 \right)$$

From which it follows that 74

$$(G_{16}(t) - G_{16}^0) e^{-(\hat{M}_{16})^{(2)} t} \leq \frac{(a_{16})^{(2)}}{(\hat{M}_{16})^{(2)}} \left[\left((\hat{P}_{16})^{(2)} + G_{17}^0 \right) e^{\left(-\frac{(\hat{P}_{16})^{(2)} + G_{17}^0}{G_{17}^0} \right)} + (\hat{P}_{16})^{(2)} \right]$$

Analogous inequalities hold also for $G_{17}, G_{18}, T_{16}, T_{17}, T_{18}$

It is now sufficient to take $\frac{(a_i)^{(1)}}{(\hat{M}_{13})^{(1)}}, \frac{(b_i)^{(1)}}{(\hat{M}_{13})^{(1)}} < 1$ and to choose 75

$(\hat{P}_{13})^{(1)}$ and $(\hat{Q}_{13})^{(1)}$ large to have

$$\frac{(a_i)^{(1)}}{(\hat{M}_{13})^{(1)}}\left[(\hat{P}_{13})^{(1)} + ((\hat{P}_{13})^{(1)} + G_j^0)e^{-\left(\frac{(\hat{P}_{13})^{(1)}+G_j^0}{G_j^0}\right)}\right] \leq (\hat{P}_{13})^{(1)}$$ 76

77

$$\frac{(b_i)^{(1)}}{(\hat{M}_{13})^{(1)}}\left[((\hat{Q}_{13})^{(1)} + T_j^0)e^{-\left(\frac{(\hat{Q}_{13})^{(1)}+T_j^0}{T_j^0}\right)} + (\hat{Q}_{13})^{(1)}\right] \leq (\hat{Q}_{13})^{(1)}$$

In order that the operator $\mathcal{A}^{(1)}$ transforms the space of sextuples of functions G_i, T_i into itself

The operator $\mathcal{A}^{(1)}$ is a contraction with respect to the metric 78

$$d\left((G^{(1)},T^{(1)}),(G^{(2)},T^{(2)})\right) =$$

$$\underset{i}{sup}\{\underset{t\in\mathbb{R}_+}{max}\left|G_i^{(1)}(t) - G_i^{(2)}(t)\right|e^{-(\hat{M}_{13})^{(1)}t}, \underset{t\in\mathbb{R}_+}{max}\left|T_i^{(1)}(t) - T_i^{(2)}(t)\right|e^{-(\hat{M}_{13})^{(1)}t}\}$$

Indeed if we denote 79

Definition of $\tilde{G},\tilde{T}$: $(\tilde{G},\tilde{T}) = \mathcal{A}^{(1)}(G,T)$

It results

$$\left|\tilde{G}_{13}^{(1)} - \tilde{G}_i^{(2)}\right| \leq \int_0^t (a_{13})^{(1)}\left|G_{14}^{(1)} - G_{14}^{(2)}\right|e^{-(\hat{M}_{13})^{(1)}s_{(13)}}e^{(\hat{M}_{13})^{(1)}s_{(13)}}\,ds_{(13)} +$$

$$\int_0^t\{(a'_{13})^{(1)}\left|G_{13}^{(1)} - G_{13}^{(2)}\right|e^{-(\hat{M}_{13})^{(1)}s_{(13)}}e^{-(\hat{M}_{13})^{(1)}s_{(13)}} +$$

$$(a''_{13})^{(1)}(T_{14}^{(1)}, s_{(13)})\left|G_{13}^{(1)} - G_{13}^{(2)}\right|e^{-(\hat{M}_{13})^{(1)}s_{(13)}}e^{(\hat{M}_{13})^{(1)}s_{(13)}} +$$

$$G_{13}^{(2)}|(a''_{13})^{(1)}(T_{14}^{(1)}, s_{(13)}) - (a''_{13})^{(1)}(T_{14}^{(2)}, s_{(13)})|\; e^{-(\hat{M}_{13})^{(1)}s_{(13)}}e^{(\hat{M}_{13})^{(1)}s_{(13)}}\}ds_{(13)}$$

Where $s_{(13)}$ represents integrand that is integrated over the interval $[0,\mathrm{t}]$

From the hypotheses on THE SOLUTIONAL EQUATIONS OF THE GOVERNING EQUATIONS it follows

$$\left|G^{(1)} - G^{(2)}\right|e^{-(\hat{M}_{13})^{(1)}t} \leq$$ 80
$$\frac{1}{(\hat{M}_{13})^{(1)}}\left((a_{13})^{(1)} + (a'_{13})^{(1)} + (\hat{A}_{13})^{(1)} + (\hat{P}_{13})^{(1)}(\hat{k}_{13})^{(1)}\right)d\left((G^{(1)},T^{(1)};\, G^{(2)},T^{(2)})\right)$$

And analogous inequalities for G_i *and* T_i. Taking into account the hypothesis (34,35,36) the result follows

Remark 1: The fact that we supposed $(a''_{13})^{(1)}$ and $(b''_{13})^{(1)}$ depending also on t can be considered as not conformal with the reality, however we have put this hypothesis ,in order that we can postulate condition necessary to prove the uniqueness of the solution bounded by $(\hat{P}_{13})^{(1)}e^{(\hat{M}_{13})^{(1)}t}$ *and* $(\hat{Q}_{13})^{(1)}e^{(\hat{M}_{13})^{(1)}t}$ respectively of $\mathbb{R}_+$.

If instead of proving the existence of the solution on $\mathbb{R}_+$, we have to prove it only on a compact then it suffices to consider that $(a_i'')^{(1)}$ and $(b_i'')^{(1)}, i = 13,14,15$ depend only on T_{14} and respectively on $G(and\ not\ on\ t)$ and hypothesis can replaced by a usual Lipschitz condition.

Remark 2: There does not exist any t where $G_i(t) = 0$ *and* $T_i(t) = 0$ 81

FROM THE GOVERING EQUATIONS AND THE CONCOMITANT DERIVED EQUATIONS IT IS

EVIDENT THAT:

$G_i\,(t) \geq G_i^0 e^{\left[-\int_0^t \{(a_i')^{(1)} - (a_i'')^{(1)}(T_{14}(s_{(13)}), s_{(13)})\} ds_{(13)}\right]} \geq 0$

$T_i\,(t) \geq T_i^0 e^{(-(b_i')^{(1)} t)} > 0$ for t > 0

Definition of $\left((\widehat{M}_{13})^{(1)}\right)_1, \left((\widehat{M}_{13})^{(1)}\right)_2$ and $\left((\widehat{M}_{13})^{(1)}\right)_3$: 82

Remark 3: if G_{13} is bounded, the same property have also G_{14} and G_{15} . indeed if

$G_{13} < (\widehat{M}_{13})^{(1)}$ it follows $\frac{dG_{14}}{dt} \leq \left((\widehat{M}_{13})^{(1)}\right)_1 - (a_{14}')^{(1)} G_{14}$ and by integrating

$G_{14} \leq \left((\widehat{M}_{13})^{(1)}\right)_2 = G_{14}^0 + 2(a_{14})^{(1)}\left((\widehat{M}_{13})^{(1)}\right)_1 / (a_{14}')^{(1)}$

In the same way , one can obtain

$G_{15} \leq \left((\widehat{M}_{13})^{(1)}\right)_3 = G_{15}^0 + 2(a_{15})^{(1)}\left((\widehat{M}_{13})^{(1)}\right)_2 / (a_{15}')^{(1)}$

If G_{14} or G_{15} is bounded, the same property follows for G_{13} , G_{15} and G_{13} , G_{14} respectively.

Remark 4: If G_{13} is bounded, from below, the same property holds for G_{14} and G_{15} . The proof is analogous with the preceding one. An analogous property is true if G_{14} is bounded from below.

Remark 5: If T_{13} is bounded from below and $\lim_{t\to\infty}((b_i'')^{(1)}\,(G(t), t)) = (b_{14}')^{(1)}$ then $T_{14} \to \infty$. 83

Definition of $(m)^{(1)}$ and ε_1 :

Indeed let t_1 be so that for $t > t_1$

$(b_{14})^{(1)} - (b_i'')^{(1)}(G(t), t) < \varepsilon_1, T_{13}\,(t) > (m)^{(1)}$

Then $\frac{dT_{14}}{dt} \geq (a_{14})^{(1)}(m)^{(1)} - \varepsilon_1 T_{14}$ which leads to 84

$T_{14} \geq \left(\frac{(a_{14})^{(1)}(m)^{(1)}}{\varepsilon_1}\right)(1 - e^{-\varepsilon_1 t}) + T_{14}^0 e^{-\varepsilon_1 t}$ If we take t such that $e^{-\varepsilon_1 t} = \frac{1}{2}$ it results

$T_{14} \geq \left(\frac{(a_{14})^{(1)}(m)^{(1)}}{2}\right), \quad t = log\frac{2}{\varepsilon_1}$ By taking now ε_1 sufficiently small one sees that T_{14} is unbounded. The same property holds for T_{15} if $\lim_{t\to\infty}(b_{15}'')^{(1)}\,(G(t), t) = (b_{15}')^{(1)}$

We now state a more precise theorem about the behaviors at infinity of the solutions of equations OF THE GOVERNING SYSTEM

It is now sufficient to take $\frac{(a_i)^{(2)}}{(\hat{M}_{16})^{(2)}}$, $\frac{(b_i)^{(2)}}{(\hat{M}_{16})^{(2)}} < 1$ and to choose

$(\hat{P}_{16})^{(2)}$ and $(\hat{Q}_{16})^{(2)}$ large to have

$$\frac{(a_i)^{(2)}}{(\hat{M}_{16})^{(2)}}\left[(\hat{P}_{16})^{(2)} + \left((\hat{P}_{16})^{(2)} + G_j^0\right)e^{-\left(\frac{(\hat{P}_{16})^{(2)} + G_j^0}{G_j^0}\right)}\right] \leq (\hat{P}_{16})^{(2)} \qquad 85$$

$$\frac{(b_i)^{(2)}}{(\hat{M}_{16})^{(2)}}\left[\left((\hat{Q}_{16})^{(2)} + T_j^0\right)e^{-\left(\frac{(\hat{Q}_{16})^{(2)} + T_j^0}{T_j^0}\right)} + (\hat{Q}_{16})^{(2)}\right] \leq (\hat{Q}_{16})^{(2)} \qquad 86$$

In order that the operator $\mathcal{A}^{(2)}$ transforms the space of sextuples of functions G_i, T_i satisfying THE

SOLUTIONAL EQUATIONS into itself

The operator $\mathcal{A}^{(2)}$ is a contraction with respect to the metric 87

$$d\left(\left((G_{19})^{(1)},(T_{19})^{(1)}\right),\left((G_{19})^{(2)},(T_{19})^{(2)}\right)\right)=$$

$$\sup_i\{\max_{t\in\mathbb{R}_+}\left|G_i^{(1)}(t)-G_i^{(2)}(t)\right|e^{-(\widehat{M}_{16})^{(2)}t},\max_{t\in\mathbb{R}_+}\left|T_i^{(1)}(t)-T_i^{(2)}(t)\right|e^{-(\widehat{M}_{16})^{(2)}t}\}$$

Indeed if we denote

Definition of $\widetilde{G_{19}},\widetilde{T_{19}}$: $\left(\widetilde{G_{19}},\widetilde{T_{19}}\right)=\mathcal{A}^{(2)}(G_{19},T_{19})$

It results 88

$$\left|\tilde{G}_{16}^{(1)}-\tilde{G}_i^{(2)}\right|\leq\int_0^t(a_{16})^{(2)}\left|G_{17}^{(1)}-G_{17}^{(2)}\right|e^{-(\widehat{M}_{16})^{(2)}s_{(16)}}e^{(\widehat{M}_{16})^{(2)}s_{(16)}}\,ds_{(16)}+$$

$$\int_0^t\{(a_{16}')^{(2)}\left|G_{16}^{(1)}-G_{16}^{(2)}\right|e^{-(\widehat{M}_{16})^{(2)}s_{(16)}}e^{-(\widehat{M}_{16})^{(2)}s_{(16)}}+$$

$$(a_{16}'')^{(2)}\left(T_{17}^{(1)},s_{(16)}\right)\left|G_{16}^{(1)}-G_{16}^{(2)}\right|e^{-(\widehat{M}_{16})^{(2)}s_{(16)}}e^{(\widehat{M}_{16})^{(2)}s_{(16)}}+$$

$$G_{16}^{(2)}|(a_{16}'')^{(2)}\left(T_{17}^{(1)},s_{(16)}\right)-(a_{16}'')^{(2)}\left(T_{17}^{(2)},s_{(16)}\right)|\ e^{-(\widehat{M}_{16})^{(2)}s_{(16)}}e^{(\widehat{M}_{16})^{(2)}s_{(16)}}\}ds_{(16)}$$

Where $s_{(16)}$ represents integrand that is integrated over the interval $[0,t]$

From the hypotheses on THE GOVERNING EQUATIONS OF THE SYSTEM it follows:

$$\left|(G_{19})^{(1)}-(G_{19})^{(2)}\right|e^{-(\widehat{M}_{16})^{(2)}t}\leq$$ 89
$$\frac{1}{(\widehat{M}_{16})^{(2)}}\left((a_{16})^{(2)}+(a_{16}')^{(2)}+(\widehat{A}_{16})^{(2)}+(\widehat{P}_{16})^{(2)}(\widehat{k}_{16})^{(2)}\right)d\left(\left((G_{19})^{(1)},(T_{19})^{(1)};\ (G_{19})^{(2)},(T_{19})^{(2)}\right)\right)$$

And analogous inequalities for G_i and T_i. Taking into account the hypothesis (34,35,36) the result follows

Remark 1: The fact that we supposed $(a_{16}'')^{(2)}$ and $(b_{16}'')^{(2)}$ depending also on t can be considered as not conformal with the reality, however we have put this hypothesis ,in order that we can postulate condition necessary to prove the uniqueness of the solution bounded by $(\widehat{P}_{16})^{(2)}e^{(\widehat{M}_{16})^{(2)}t}$ and $(\widehat{Q}_{16})^{(2)}e^{(\widehat{M}_{16})^{(2)}t}$ respectively of $\mathbb{R}_+$. 90

If instead of proving the existence of the solution on $\mathbb{R}_+$, we have to prove it only on a compact then it suffices to consider that $(a_i'')^{(2)}$ and $(b_i'')^{(2)}, i=16,17,18$ depend only on T_{17} and respectively on (G_{19})(and not on t) and hypothesis can replaced by a usual Lipschitz condition.

Remark 2: There does not exist any t where $G_i(t)=0$ and $T_i(t)=0$ 91

From THE SYSTEMIC GOVERNING EQUSATIONS ,IT RELUTS THAT:

$$G_i(t)\geq G_i^0e^{\left[-\int_0^t\{(a_i')^{(2)}-(a_i'')^{(2)}(T_{17}(s_{(16)}),s_{(16)})\}ds_{(16)}\right]}\geq 0$$

$$T_i(t)\geq T_i^0e^{\left(-(b_i')^{(2)}t\right)}>0\quad\text{for } t>0$$

Definition of $\left((\widehat{M}_{16})^{(2)}\right)_1,\left((\widehat{M}_{16})^{(2)}\right)_2$ and $\left((\widehat{M}_{16})^{(2)}\right)_3$: 92

Remark 3: if G_{16} is bounded, the same property have also G_{17} and G_{18} . indeed if

$G_{16}<(\widehat{M}_{16})^{(2)}$ it follows $\frac{dG_{17}}{dt}\leq\left((\widehat{M}_{16})^{(2)}\right)_1-(a_{17}')^{(2)}G_{17}$ and by integrating

$$G_{17}\leq\left((\widehat{M}_{16})^{(2)}\right)_2=G_{17}^0+2(a_{17})^{(2)}\left((\widehat{M}_{16})^{(2)}\right)_1/(a_{17}')^{(2)}$$

In the same way , one can obtain

$G_{18} \leq \left((\widehat{M}_{16})^{(2)}\right)_3 = G_{18}^0 + 2(a_{18})^{(2)}\left((\widehat{M}_{16})^{(2)}\right)_2/(a'_{18})^{(2)}$

If G_{17} or G_{18} is bounded, the same property follows for G_{16} , G_{18} and G_{16} , G_{17} respectively.

Remark 4: If G_{16} is bounded, from below, the same property holds for G_{17} and G_{18} . The proof is analogous with the preceding one. An analogous property is true if G_{17} is bounded from below.

Remark 5: If T_{16} is bounded from below and $\lim_{t\to\infty}((b''_i)^{(2)}((G_{19})(t),t)) = (b'_{17})^{(2)}$ then $T_{17} \to \infty$. 93

Definition of $(m)^{(2)}$ and ε_2 :

Indeed let t_2 be so that for $t > t_2$

$(b_{17})^{(2)} - (b''_i)^{(2)}((G_{19})(t),t) < \varepsilon_2, T_{16}(t) > (m)^{(2)}$

Then $\frac{dT_{17}}{dt} \geq (a_{17})^{(2)}(m)^{(2)} - \varepsilon_2 T_{17}$ which leads to 94

$T_{17} \geq \left(\frac{(a_{17})^{(2)}(m)^{(2)}}{\varepsilon_2}\right)(1-e^{-\varepsilon_2 t}) + T_{17}^0 e^{-\varepsilon_2 t}$ If we take t such that $e^{-\varepsilon_2 t} = \frac{1}{2}$ it results

$T_{17} \geq \left(\frac{(a_{17})^{(2)}(m)^{(2)}}{2}\right), \quad t = \log\frac{2}{\varepsilon_2}$ By taking now ε_2 sufficiently small one sees that T_{17} is unbounded. 95
The same property holds for T_{18} if $\lim_{t\to\infty}(b''_{18})^{(2)}\left((G_{19})(t),t\right) = (b'_{18})^{(2)}$

We now state a more precise theorem about the behaviors at infinity of the solutions of equations 37 to 42

Behavior of the solutions of GOVERNING equations: 96

Theorem 2: If we denote and define

Definition of $(\sigma_1)^{(1)}, (\sigma_2)^{(1)}, (\tau_1)^{(1)}, (\tau_2)^{(1)}$:

(a) $\sigma_1)^{(1)}, (\sigma_2)^{(1)}, (\tau_1)^{(1)}, (\tau_2)^{(1)}$ four constants satisfying

$-(\sigma_2)^{(1)} \leq -(a'_{13})^{(1)} + (a'_{14})^{(1)} - (a''_{13})^{(1)}(T_{14},t) + (a''_{14})^{(1)}(T_{14},t) \leq -(\sigma_1)^{(1)}$

$-(\tau_2)^{(1)} \leq -(b'_{13})^{(1)} + (b'_{14})^{(1)} - (b''_{13})^{(1)}(G,t) - (b''_{14})^{(1)}(G,t) \leq -(\tau_1)^{(1)}$

97

Definition of $(\nu_1)^{(1)}, (\nu_2)^{(1)}, (u_1)^{(1)}, (u_2)^{(1)}, \nu^{(1)}, u^{(1)}$:

(b) By $(\nu_1)^{(1)} > 0, (\nu_2)^{(1)} < 0$ and respectively $(u_1)^{(1)} > 0, (u_2)^{(1)} < 0$ the roots of the equations $(a_{14})^{(1)}\left(\nu^{(1)}\right)^2 + (\sigma_1)^{(1)}\nu^{(1)} - (a_{13})^{(1)} = 0$ and $(b_{14})^{(1)}\left(u^{(1)}\right)^2 + (\tau_1)^{(1)}u^{(1)} - (b_{13})^{(1)} = 0$

Definition of $(\bar{\nu}_1)^{(1)},, (\bar{\nu}_2)^{(1)}, (\bar{u}_1)^{(1)}, (\bar{u}_2)^{(1)}$: 98

By $(\bar{\nu}_1)^{(1)} > 0, (\bar{\nu}_2)^{(1)} < 0$ and respectively $(\bar{u}_1)^{(1)} > 0, (\bar{u}_2)^{(1)} < 0$ the roots of the equations $(a_{14})^{(1)}\left(\nu^{(1)}\right)^2 + (\sigma_2)^{(1)}\nu^{(1)} - (a_{13})^{(1)} = 0$ and $(b_{14})^{(1)}\left(u^{(1)}\right)^2 + (\tau_2)^{(1)}u^{(1)} - (b_{13})^{(1)} = 0$

Definition of $(m_1)^{(1)}, (m_2)^{(1)}, (\mu_1)^{(1)}, (\mu_2)^{(1)}, (\nu_0)^{(1)}$:- 99

(c) If we define $(m_1)^{(1)}, (m_2)^{(1)}, (\mu_1)^{(1)}, (\mu_2)^{(1)}$ by

$(m_2)^{(1)} = (\nu_0)^{(1)}, (m_1)^{(1)} = (\nu_1)^{(1)}, \ if \ (\nu_0)^{(1)} < (\nu_1)^{(1)}$

$(m_2)^{(1)} = (\nu_1)^{(1)}, (m_1)^{(1)} = (\bar{\nu}_1)^{(1)}, if\ (\nu_1)^{(1)} < (\nu_0)^{(1)} < (\bar{\nu}_1)^{(1)},$

and $\boxed{(\nu_0)^{(1)} = \frac{G_{13}^0}{G_{14}^0}}$

$(m_2)^{(1)} = (\nu_1)^{(1)}, (m_1)^{(1)} = (\nu_0)^{(1)},\ if\ (\bar{\nu}_1)^{(1)} < (\nu_0)^{(1)}$

and analogously 100

$(\mu_2)^{(1)} = (u_0)^{(1)}, (\mu_1)^{(1)} = (u_1)^{(1)},\ if\ (u_0)^{(1)} < (u_1)^{(1)}$

$(\mu_2)^{(1)} = (u_1)^{(1)}, (\mu_1)^{(1)} = (\bar{u}_1)^{(1)}, if\ (u_1)^{(1)} < (u_0)^{(1)} < (\bar{u}_1)^{(1)},$

and $\boxed{(u_0)^{(1)} = \frac{T_{13}^0}{T_{14}^0}}$

$(\mu_2)^{(1)} = (u_1)^{(1)}, (\mu_1)^{(1)} = (u_0)^{(1)}, if\ (\bar{u}_1)^{(1)} < (u_0)^{(1)}$ where $(u_1)^{(1)}, (\bar{u}_1)^{(1)}$ are defined IN THE FOREGOING

Then the solution of GOVERNING EQUATIONS satisfies the inequalities

$$G_{13}^0 e^{((S_1)^{(1)}-(p_{13})^{(1)})t} \le G_{13}(t) \le G_{13}^0 e^{(S_1)^{(1)}t}$$

where $(p_i)^{(1)}$ is defined IN THE PREVIOUS MODULE

$$\frac{1}{(m_1)^{(1)}} G_{13}^0 e^{((S_1)^{(1)}-(p_{13})^{(1)})t} \le G_{14}(t) \le \frac{1}{(m_2)^{(1)}} G_{13}^0 e^{(S_1)^{(1)}t}$$

$$\left(\frac{(a_{15})^{(1)} G_{13}^0}{(m_1)^{(1)}((S_1)^{(1)}-(p_{13})^{(1)}-(S_2)^{(1)})}\left[e^{((S_1)^{(1)}-(p_{13})^{(1)})t} - e^{-(S_2)^{(1)}t}\right] + G_{15}^0 e^{-(S_2)^{(1)}t} \le G_{15}(t) \le \right.$$
$$\left.\frac{(a_{15})^{(1)} G_{13}^0}{(m_2)^{(1)}((S_1)^{(1)}-(a_{15}')^{(1)})}\left[e^{(S_1)^{(1)}t} - e^{-(a_{15}')^{(1)}t}\right] + G_{15}^0 e^{-(a_{15}')^{(1)}t}\right)$$

$$\boxed{T_{13}^0 e^{(R_1)^{(1)}t} \le T_{13}(t) \le T_{13}^0 e^{((R_1)^{(1)}+(r_{13})^{(1)})t}}$$

$$\frac{1}{(\mu_1)^{(1)}} T_{13}^0 e^{(R_1)^{(1)}t} \le T_{13}(t) \le \frac{1}{(\mu_2)^{(1)}} T_{13}^0 e^{((R_1)^{(1)}+(r_{13})^{(1)})t}$$

$$\frac{(b_{15})^{(1)} T_{13}^0}{(\mu_1)^{(1)}((R_1)^{(1)}-(b_{15}')^{(1)})}\left[e^{(R_1)^{(1)}t} - e^{-(b_{15}')^{(1)}t}\right] + T_{15}^0 e^{-(b_{15}')^{(1)}t} \le T_{15}(t) \le$$ 101

$$\frac{(a_{15})^{(1)} T_{13}^0}{(\mu_2)^{(1)}((R_1)^{(1)}+(r_{13})^{(1)}+(R_2)^{(1)})}\left[e^{((R_1)^{(1)}+(r_{13})^{(1)})t} - e^{-(R_2)^{(1)}t}\right] + T_{15}^0 e^{-(R_2)^{(1)}t}$$

Definition of $(S_1)^{(1)}, (S_2)^{(1)}, (R_1)^{(1)}, (R_2)^{(1)}$:- 102

Where $(S_1)^{(1)} = (a_{13})^{(1)}(m_2)^{(1)} - (a_{13}')^{(1)}$

$(S_2)^{(1)} = (a_{15})^{(1)} - (p_{15})^{(1)}$

$(R_1)^{(1)} = (b_{13})^{(1)}(\mu_2)^{(1)} - (b_{13}')^{(1)}$

$(R_2)^{(1)} = (b_{15}')^{(1)} - (r_{15})^{(1)}$

Behavior of the SOLUTIONAL EQUATIONS

Theorem 2: If we denote and define

Definition of $(\sigma_1)^{(2)}, (\sigma_2)^{(2)}, (\tau_1)^{(2)}, (\tau_2)^{(2)}$: 103

(d) $\sigma_1)^{(2)}, (\sigma_2)^{(2)}, (\tau_1)^{(2)}, (\tau_2)^{(2)}$ four constants satisfying

$$-(\sigma_2)^{(2)} \leq -(a'_{16})^{(2)} + (a'_{17})^{(2)} - (a''_{16})^{(2)}(\mathrm{T}_{17}, t) + (a''_{17})^{(2)}(\mathrm{T}_{17}, t) \leq -(\sigma_1)^{(2)}$$

$$-(\tau_2)^{(2)} \leq -(b'_{16})^{(2)} + (b'_{17})^{(2)} - (b''_{16})^{(2)}\big((G_{19}), t\big) - (b''_{17})^{(2)}\big((G_{19}), t\big) \leq -(\tau_1)^{(2)}$$

Definition of $(\nu_1)^{(2)}, (\nu_2)^{(2)}, (u_1)^{(2)}, (u_2)^{(2)}$: 104

By $(\nu_1)^{(2)} > 0, (\nu_2)^{(2)} < 0$ and respectively $(u_1)^{(2)} > 0, (u_2)^{(2)} < 0$ the roots

(e) of the equations $(a_{17})^{(2)}\big(\nu^{(2)}\big)^2 + (\sigma_1)^{(2)}\nu^{(2)} - (a_{16})^{(2)} = 0$

and $(b_{14})^{(2)}\big(u^{(2)}\big)^2 + (\tau_1)^{(2)}u^{(2)} - (b_{16})^{(2)} = 0$ and

Definition of $(\bar{\nu}_1)^{(2)},, (\bar{\nu}_2)^{(2)}, (\bar{u}_1)^{(2)}, (\bar{u}_2)^{(2)}$: 105

By $(\bar{\nu}_1)^{(2)} > 0, (\bar{\nu}_2)^{(2)} < 0$ and respectively $(\bar{u}_1)^{(2)} > 0, (\bar{u}_2)^{(2)} < 0$ the

roots of the equations $(a_{17})^{(2)}\big(\nu^{(2)}\big)^2 + (\sigma_2)^{(2)}\nu^{(2)} - (a_{16})^{(2)} = 0$

and $(b_{17})^{(2)}\big(u^{(2)}\big)^2 + (\tau_2)^{(2)}u^{(2)} - (b_{16})^{(2)} = 0$

Definition of $(m_1)^{(2)}, (m_2)^{(2)}, (\mu_1)^{(2)}, (\mu_2)^{(2)}$:- 106

(f) If we define $(m_1)^{(2)}, (m_2)^{(2)}, (\mu_1)^{(2)}, (\mu_2)^{(2)}$ by

$(m_2)^{(2)} = (\nu_0)^{(2)}, (m_1)^{(2)} = (\nu_1)^{(2)},\ \boldsymbol{if}\ (\nu_0)^{(2)} < (\nu_1)^{(2)}$

$(m_2)^{(2)} = (\nu_1)^{(2)}, (m_1)^{(2)} = (\bar{\nu}_1)^{(2)},\ \boldsymbol{if}\ (\nu_1)^{(2)} < (\nu_0)^{(2)} < (\bar{\nu}_1)^{(2)},$

and $\boxed{(\nu_0)^{(2)} = \frac{\mathrm{G}_{16}^0}{\mathrm{G}_{17}^0}}$

$(m_2)^{(2)} = (\nu_1)^{(2)}, (m_1)^{(2)} = (\nu_0)^{(2)},\ \boldsymbol{if}\ (\bar{\nu}_1)^{(2)} < (\nu_0)^{(2)}$

and analogously

$(\mu_2)^{(2)} = (u_0)^{(2)}, (\mu_1)^{(2)} = (u_1)^{(2)},\ \boldsymbol{if}\ (u_0)^{(2)} < (u_1)^{(2)}$

$(\mu_2)^{(2)} = (u_1)^{(2)}, (\mu_1)^{(2)} = (\bar{u}_1)^{(2)},\ \boldsymbol{if}\ (u_1)^{(2)} < (u_0)^{(2)} < (\bar{u}_1)^{(2)},$

and $\boxed{(u_0)^{(2)} = \frac{\mathrm{T}_{16}^0}{\mathrm{T}_{17}^0}}$

$(\mu_2)^{(2)} = (u_1)^{(2)}, (\mu_1)^{(2)} = (u_0)^{(2)},\ \boldsymbol{if}\ (\bar{u}_1)^{(2)} < (u_0)^{(2)}$

Then the solution of GOVERNING EQUATIONS OF THE HOLISTIC AND UNIVERSALISED SYSTEM satisfies the inequalities

$$\mathrm{G}_{16}^0 \mathrm{e}^{((\mathrm{S}_1)^{(2)} - (p_{16})^{(2)})\mathrm{t}} \leq G_{16}(t) \leq \mathrm{G}_{16}^0 \mathrm{e}^{(\mathrm{S}_1)^{(2)}\mathrm{t}}$$

$(p_i)^{(2)}$ is defined ABOVE

$$\frac{1}{(m_1)^{(2)}}\mathrm{G}_{16}^0 \mathrm{e}^{((\mathrm{S}_1)^{(2)}-(p_{16})^{(2)})\mathrm{t}} \leq G_{17}(t) \leq \frac{1}{(m_2)^{(2)}}\mathrm{G}_{16}^0 \mathrm{e}^{(\mathrm{S}_1)^{(2)}\mathrm{t}}$$

$$\Big(\frac{(a_{18})^{(2)}\mathrm{G}_{16}^0}{(m_1)^{(2)}((\mathrm{S}_1)^{(2)}-(p_{16})^{(2)}-(\mathrm{S}_2)^{(2)})}\left[\mathrm{e}^{((\mathrm{S}_1)^{(2)}-(p_{16})^{(2)})\mathrm{t}} - \mathrm{e}^{-(\mathrm{S}_2)^{(2)}\mathrm{t}}\right] + \mathrm{G}_{18}^0 \mathrm{e}^{-(\mathrm{S}_2)^{(2)}\mathrm{t}} \leq \mathrm{G}_{18}(t) \leq$$
$$\frac{(a_{18})^{(2)}\mathrm{G}_{16}^0}{(m_2)^{(2)}((\mathrm{S}_1)^{(2)}-(a_{18}')^{(2)})}[\mathrm{e}^{(\mathrm{S}_1)^{(2)}\mathrm{t}} - \mathrm{e}^{-(a_{18}')^{(2)}\mathrm{t}}] + \mathrm{G}_{18}^0 \mathrm{e}^{-(a_{18}')^{(2)}\mathrm{t}}\Big)$$

$$\boxed{\mathrm{T}_{16}^0 \mathrm{e}^{(\mathrm{R}_1)^{(2)}t} \leq T_{16}(t) \leq \mathrm{T}_{16}^0 \mathrm{e}^{((\mathrm{R}_1)^{(2)}+(r_{16})^{(2)})t}}$$

$$\frac{1}{(\mu_1)^{(2)}}\mathrm{T}_{16}^0 \mathrm{e}^{(\mathrm{R}_1)^{(2)}t} \leq T_{16}(t) \leq \frac{1}{(\mu_2)^{(2)}}\mathrm{T}_{16}^0 \mathrm{e}^{((\mathrm{R}_1)^{(2)}+(r_{16})^{(2)})t}$$

$$\frac{(b_{18})^{(2)}\mathrm{T}_{16}^0}{(\mu_1)^{(2)}((\mathrm{R}_1)^{(2)}-(b_{18}')^{(2)})}\left[\mathrm{e}^{(\mathrm{R}_1)^{(2)}t} - \mathrm{e}^{-(b_{18}')^{(2)}t}\right] + \mathrm{T}_{18}^0 \mathrm{e}^{-(b_{18}')^{(2)}t} \leq T_{18}(t) \leq$$

$$\frac{(a_{18})^{(2)}\mathrm{T}_{16}^0}{(\mu_2)^{(2)}((\mathrm{R}_1)^{(2)}+(r_{16})^{(2)}+(\mathrm{R}_2)^{(2)})}\left[\mathrm{e}^{((\mathrm{R}_1)^{(2)}+(r_{16})^{(2)})t} - \mathrm{e}^{-(\mathrm{R}_2)^{(2)}t}\right] + \mathrm{T}_{18}^0 \mathrm{e}^{-(\mathrm{R}_2)^{(2)}t}$$

Definition of $(\mathrm{S}_1)^{(2)}, (\mathrm{S}_2)^{(2)}, (\mathrm{R}_1)^{(2)}, (\mathrm{R}_2)^{(2)}$:-

Where $(\mathrm{S}_1)^{(2)} = (a_{16})^{(2)}(m_2)^{(2)} - (a_{16}')^{(2)}$ 107

$(\mathrm{S}_2)^{(2)} = (a_{18})^{(2)} - (p_{18})^{(2)}$

$(R_1)^{(2)} = (b_{16})^{(2)}(\mu_2)^{(1)} - (b_{16}')^{(2)}$

$(\mathrm{R}_2)^{(2)} = (b_{18}')^{(2)} - (r_{18})^{(2)}$

PROOF : From THE GOVERNING EQUATIONS OF THE GLOBAL SYTEMAL EQUATIONS we obtain 108

$$\frac{d\nu^{(1)}}{dt} = (a_{13})^{(1)} - \left((a_{13}')^{(1)} - (a_{14}')^{(1)} + (a_{13}'')^{(1)}(T_{14},t)\right) - (a_{14}'')^{(1)}(T_{14},t)\nu^{(1)} - (a_{14})^{(1)}\nu^{(1)}$$

Definition of $\nu^{(1)}$:- $\boxed{\nu^{(1)} = \frac{G_{13}}{G_{14}}}$

It follows

$$-\left((a_{14})^{(1)}\left(\nu^{(1)}\right)^2 + (\sigma_2)^{(1)}\nu^{(1)} - (a_{13})^{(1)}\right) \leq \frac{d\nu^{(1)}}{dt} \leq -\left((a_{14})^{(1)}\left(\nu^{(1)}\right)^2 + (\sigma_1)^{(1)}\nu^{(1)} - (a_{13})^{(1)}\right)$$

From which one obtains

Definition of $(\bar{\nu}_1)^{(1)}, (\nu_0)^{(1)}$:-

(a) For $0 < \boxed{(\nu_0)^{(1)} = \frac{G_{13}^0}{G_{14}^0}} < (\nu_1)^{(1)} < (\bar{\nu}_1)^{(1)}$

$$\nu^{(1)}(t) \geq \frac{(\nu_1)^{(1)} + (C)^{(1)}(\nu_2)^{(1)} e^{\left[-(a_{14})^{(1)}\left((\nu_1)^{(1)}-(\nu_0)^{(1)}\right)t\right]}}{1+(C)^{(1)} e^{\left[-(a_{14})^{(1)}\left((\nu_1)^{(1)}-(\nu_0)^{(1)}\right)t\right]}} \quad , \quad \boxed{(C)^{(1)} = \frac{(\nu_1)^{(1)}-(\nu_0)^{(1)}}{(\nu_0)^{(1)}-(\nu_2)^{(1)}}}$$

it follows $(\nu_0)^{(1)} \leq \nu^{(1)}(t) \leq (\nu_1)^{(1)}$

In the same manner , we get 109

$$\nu^{(1)}(t) \leq \frac{(\bar{\nu}_1)^{(1)}+(\bar{C})^{(1)}(\bar{\nu}_2)^{(1)}e^{\left[-(a_{14})^{(1)}\left((\bar{\nu}_1)^{(1)}-(\bar{\nu}_2)^{(1)}\right)t\right]}}{1+(\bar{C})^{(1)}e^{\left[-(a_{14})^{(1)}\left((\bar{\nu}_1)^{(1)}-(\bar{\nu}_2)^{(1)}\right)t\right]}} \quad , \quad \boxed{(\bar{C})^{(1)} = \frac{(\bar{\nu}_1)^{(1)}-(\nu_0)^{(1)}}{(\nu_0)^{(1)}-(\bar{\nu}_2)^{(1)}}}$$

From which we deduce $(\nu_0)^{(1)} \leq \nu^{(1)}(t) \leq (\bar{\nu}_1)^{(1)}$

(b) If $0 < (\nu_1)^{(1)} < (\nu_0)^{(1)} = \frac{G_{13}^0}{G_{14}^0} < (\bar{\nu}_1)^{(1)}$ we find like in the previous case, 110

$$(\nu_1)^{(1)} \leq \frac{(\nu_1)^{(1)}+(C)^{(1)}(\nu_2)^{(1)}e^{\left[-(a_{14})^{(1)}\left((\nu_1)^{(1)}-(\nu_2)^{(1)}\right)t\right]}}{1+(C)^{(1)}e^{\left[-(a_{14})^{(1)}\left((\nu_1)^{(1)}-(\nu_2)^{(1)}\right)t\right]}} \leq \nu^{(1)}(t) \leq$$

$$\frac{(\bar{\nu}_1)^{(1)}+(\bar{C})^{(1)}(\bar{\nu}_2)^{(1)}e^{\left[-(a_{14})^{(1)}\left((\bar{\nu}_1)^{(1)}-(\bar{\nu}_2)^{(1)}\right)t\right]}}{1+(\bar{C})^{(1)}e^{\left[-(a_{14})^{(1)}\left((\bar{\nu}_1)^{(1)}-(\bar{\nu}_2)^{(1)}\right)t\right]}} \leq (\bar{\nu}_1)^{(1)}$$

(c) If $0 < (\nu_1)^{(1)} \leq (\bar{\nu}_1)^{(1)} \leq \boxed{(\nu_0)^{(1)} = \frac{G_{13}^0}{G_{14}^0}}$, we obtain 111

$$(\nu_1)^{(1)} \leq \nu^{(1)}(t) \leq \frac{(\bar{\nu}_1)^{(1)}+(\bar{C})^{(1)}(\bar{\nu}_2)^{(1)}e^{\left[-(a_{14})^{(1)}\left((\bar{\nu}_1)^{(1)}-(\bar{\nu}_2)^{(1)}\right)t\right]}}{1+(\bar{C})^{(1)}e^{\left[-(a_{14})^{(1)}\left((\bar{\nu}_1)^{(1)}-(\bar{\nu}_2)^{(1)}\right)t\right]}} \leq (\nu_0)^{(1)}$$

And so with the notation of the first part of condition (c) , we have

Definition of $\nu^{(1)}(t)$:-

$$(m_2)^{(1)} \leq \nu^{(1)}(t) \leq (m_1)^{(1)}, \quad \boxed{\nu^{(1)}(t) = \frac{G_{13}(t)}{G_{14}(t)}}$$

In a completely analogous way, we obtain

Definition of $u^{(1)}(t)$:-

$$(\mu_2)^{(1)} \leq u^{(1)}(t) \leq (\mu_1)^{(1)}, \quad \boxed{u^{(1)}(t) = \frac{T_{13}(t)}{T_{14}(t)}}$$

Now, using this result and replacing it in 19, 20,21,22,23, and 24 we get easily the result stated in the theorem.

Particular case :

If $(a''_{13})^{(1)} = (a''_{14})^{(1)}, then\ (\sigma_1)^{(1)} = (\sigma_2)^{(1)}$ and in this case $(\nu_1)^{(1)} = (\bar{\nu}_1)^{(1)}$ if in addition $(\nu_0)^{(1)} = (\nu_1)^{(1)}$ then $\nu^{(1)}(t) = (\nu_0)^{(1)}$ and as a consequence $G_{13}(t) = (\nu_0)^{(1)}G_{14}(t)$ this also defines $(\nu_0)^{(1)}$ for the special case

Analogously if $(b''_{13})^{(1)} = (b''_{14})^{(1)}, then\ (\tau_1)^{(1)} = (\tau_2)^{(1)}$ and then

$(u_1)^{(1)} = (\bar{u}_1)^{(1)}$if in addition $(u_0)^{(1)} = (u_1)^{(1)}$ then $T_{13}(t) = (u_0)^{(1)}T_{14}(t)$ This is an important consequence of the relation between $(\nu_1)^{(1)}$ and $(\bar{\nu}_1)^{(1)}$, and definition of $(u_0)^{(1)}$.

PROOF : From THE GOVERSING EQUATIONS OF GLOBAL SYSTEM we obtain

$$\frac{d\nu^{(2)}}{dt} = (a_{16})^{(2)} - \left((a'_{16})^{(2)} - (a'_{17})^{(2)} + (a''_{16})^{(2)}(T_{17},t)\right) - (a''_{17})^{(2)}(T_{17},t)\nu^{(2)} - (a_{17})^{(2)}\nu^{(2)}$$

Definition of $\nu^{(2)}$:- $\boxed{\nu^{(2)} = \frac{G_{16}}{G_{17}}}$

It follows

$$-\left((a_{17})^{(2)}\left(\nu^{(2)}\right)^2+(\sigma_2)^{(2)}\nu^{(2)}-(a_{16})^{(2)}\right)\leq\frac{d\nu^{(2)}}{dt}\leq-\left((a_{17})^{(2)}\left(\nu^{(2)}\right)^2+(\sigma_1)^{(2)}\nu^{(2)}-(a_{16})^{(2)}\right)$$

From which one obtains 112

Definition of $(\bar{\nu}_1)^{(2)},(\nu_0)^{(2)}$:-

(d) For $0<(\nu_0)^{(2)}=\frac{G_{16}^0}{G_{17}^0}<(\nu_1)^{(2)}<(\bar{\nu}_1)^{(2)}$

$$\nu^{(2)}(t)\geq\frac{(\nu_1)^{(2)}+(C)^{(2)}(\nu_2)^{(2)}e^{\left[-(a_{17})^{(2)}\left((\nu_1)^{(2)}-(\nu_0)^{(2)}\right)t\right]}}{1+(C)^{(2)}e^{\left[-(a_{17})^{(2)}\left((\nu_1)^{(2)}-(\nu_0)^{(2)}\right)t\right]}}\quad,\quad\boxed{(C)^{(2)}=\frac{(\nu_1)^{(2)}-(\nu_0)^{(2)}}{(\nu_0)^{(2)}-(\nu_2)^{(2)}}}$$

it follows $(\nu_0)^{(2)}\leq\nu^{(2)}(t)\leq(\nu_1)^{(2)}$

In the same manner , we get

$$\nu^{(2)}(t)\leq\frac{(\bar{\nu}_1)^{(2)}+(\bar{C})^{(2)}(\bar{\nu}_2)^{(2)}e^{\left[-(a_{17})^{(2)}\left((\bar{\nu}_1)^{(2)}-(\bar{\nu}_2)^{(2)}\right)t\right]}}{1+(\bar{C})^{(2)}e^{\left[-(a_{17})^{(2)}\left((\bar{\nu}_1)^{(2)}-(\bar{\nu}_2)^{(2)}\right)t\right]}}\quad,\quad\boxed{(\bar{C})^{(2)}=\frac{(\bar{\nu}_1)^{(2)}-(\nu_0)^{(2)}}{(\nu_0)^{(2)}-(\bar{\nu}_2)^{(2)}}}$$

From which we deduce $(\nu_0)^{(2)}\leq\nu^{(2)}(t)\leq(\bar{\nu}_1)^{(2)}$

(e) If $0<(\nu_1)^{(2)}<(\nu_0)^{(2)}=\frac{G_{16}^0}{G_{17}^0}<(\bar{\nu}_1)^{(2)}$ we find like in the previous case,

$$(\nu_1)^{(2)}\leq\frac{(\nu_1)^{(2)}+(C)^{(2)}(\nu_2)^{(2)}e^{\left[-(a_{17})^{(2)}\left((\nu_1)^{(2)}-(\nu_2)^{(2)}\right)t\right]}}{1+(C)^{(2)}e^{\left[-(a_{17})^{(2)}\left((\nu_1)^{(2)}-(\nu_2)^{(2)}\right)t\right]}}\leq\nu^{(2)}(t)\leq$$

$$\frac{(\bar{\nu}_1)^{(2)}+(\bar{C})^{(2)}(\bar{\nu}_2)^{(2)}e^{\left[-(a_{17})^{(2)}\left((\bar{\nu}_1)^{(2)}-(\bar{\nu}_2)^{(2)}\right)t\right]}}{1+(\bar{C})^{(2)}e^{\left[-(a_{17})^{(2)}\left((\bar{\nu}_1)^{(2)}-(\bar{\nu}_2)^{(2)}\right)t\right]}}\leq(\bar{\nu}_1)^{(2)}$$

(f) If $0<(\nu_1)^{(2)}\leq(\bar{\nu}_1)^{(2)}\leq(\nu_0)^{(2)}=\frac{G_{16}^0}{G_{17}^0}$, we obtain 113

$$(\nu_1)^{(2)}\leq\nu^{(2)}(t)\leq\frac{(\bar{\nu}_1)^{(2)}+(\bar{C})^{(2)}(\bar{\nu}_2)^{(2)}e^{\left[-(a_{17})^{(2)}\left((\bar{\nu}_1)^{(2)}-(\bar{\nu}_2)^{(2)}\right)t\right]}}{1+(\bar{C})^{(2)}e^{\left[-(a_{17})^{(2)}\left((\bar{\nu}_1)^{(2)}-(\bar{\nu}_2)^{(2)}\right)t\right]}}\leq(\nu_0)^{(2)}$$

And so with the notation of the first part of condition (c) , we have

Definition of $\nu^{(2)}(t)$:- 114

$$(m_2)^{(2)}\leq\nu^{(2)}(t)\leq(m_1)^{(2)},\quad\boxed{\nu^{(2)}(t)=\frac{G_{16}(t)}{G_{17}(t)}}$$

In a completely analogous way, we obtain 115

Definition of $u^{(2)}(t)$:-

$$(\mu_2)^{(2)}\leq u^{(2)}(t)\leq(\mu_1)^{(2)},\quad\boxed{u^{(2)}(t)=\frac{T_{16}(t)}{T_{17}(t)}}$$

Now, using this result and replacing it in THE SOLUTIONS OF THE UNIVERSALISTIC SYSTEMAL EQUATSIONS we get easily the result stated in the theorem.

Particular case : 116

If $(a_{16}'')^{(2)}=(a_{17}'')^{(2)}, then$ $(\sigma_1)^{(2)}=(\sigma_2)^{(2)}$ and in this case $(\nu_1)^{(2)}=(\bar{\nu}_1)^{(2)}$ if in addition $(\nu_0)^{(2)}=(\nu_1)^{(2)}$ then $\nu^{(2)}(t)=(\nu_0)^{(2)}$ and as a consequence $G_{16}(t)=(\nu_0)^{(2)}G_{17}(t)$

Analogously if $(b''_{16})^{(2)} = (b''_{17})^{(2)}, then$ $(\tau_1)^{(2)} = (\tau_2)^{(2)}$ and then

$(u_1)^{(2)} = (\bar{u}_1)^{(2)}$if in addition $(u_0)^{(2)} = (u_1)^{(2)}$ then $T_{16}(t) = (u_0)^{(2)}T_{17}(t)$ This is an important consequence of the relation between $(\nu_1)^{(2)}$ and $(\bar{\nu}_1)^{(2)}$

We can prove the following 117

<u>Theorem 3</u>: If $(a''_i)^{(1)} and$ $(b''_i)^{(1)}$ are independent on t , and the conditions

$(a'_{13})^{(1)}(a'_{14})^{(1)} - (a_{13})^{(1)}(a_{14})^{(1)} < 0$

$(a'_{13})^{(1)}(a'_{14})^{(1)} - (a_{13})^{(1)}(a_{14})^{(1)} + (a_{13})^{(1)}(p_{13})^{(1)} + (a'_{14})^{(1)}(p_{14})^{(1)} + (p_{13})^{(1)}(p_{14})^{(1)} > 0$

$(b'_{13})^{(1)}(b'_{14})^{(1)} - (b_{13})^{(1)}(b_{14})^{(1)} > 0$,

$(b'_{13})^{(1)}(b'_{14})^{(1)} - (b_{13})^{(1)}(b_{14})^{(1)} - (b'_{13})^{(1)}(r_{14})^{(1)} - (b'_{14})^{(1)}(r_{14})^{(1)} + (r_{13})^{(1)}(r_{14})^{(1)} < 0$

$with$ $(p_{13})^{(1)}, (r_{14})^{(1)}$ as defined by equation 25 are satisfied , then the system

If $(a''_i)^{(2)} and$ $(b''_i)^{(2)}$ are independent on t , and the conditions

$(a'_{16})^{(2)}(a'_{17})^{(2)} - (a_{16})^{(2)}(a_{17})^{(2)} < 0$

$(a'_{16})^{(2)}(a'_{17})^{(2)} - (a_{16})^{(2)}(a_{17})^{(2)} + (a_{16})^{(2)}(p_{16})^{(2)} + (a'_{17})^{(2)}(p_{17})^{(2)} + (p_{16})^{(2)}(p_{17})^{(2)} > 0$

$(b'_{16})^{(2)}(b'_{17})^{(2)} - (b_{16})^{(2)}(b_{17})^{(2)} > 0$,

$(b'_{16})^{(2)}(b'_{17})^{(2)} - (b_{16})^{(2)}(b_{17})^{(2)} - (b'_{16})^{(2)}(r_{17})^{(2)} - (b'_{17})^{(2)}(r_{17})^{(2)} + (r_{16})^{(2)}(r_{17})^{(2)} < 0$

$with$ $(p_{16})^{(2)}, (r_{17})^{(2)}$ DEFINED AS ABOVE, then the system

$(a_{13})^{(1)}G_{14} - \left[(a'_{13})^{(1)} + (a''_{13})^{(1)}(T_{14})\right]G_{13} = 0$ 118

$(a_{14})^{(1)}G_{13} - \left[(a'_{14})^{(1)} + (a''_{14})^{(1)}(T_{14})\right]G_{14} = 0$ 119

$(a_{15})^{(1)}G_{14} - \left[(a'_{15})^{(1)} + (a''_{15})^{(1)}(T_{14})\right]G_{15} = 0$ 120

$(b_{13})^{(1)}T_{14} - [(b'_{13})^{(1)} - (b''_{13})^{(1)}(G)\ ]T_{13} = 0$ 121

$(b_{14})^{(1)}T_{13} - [(b'_{14})^{(1)} - (b''_{14})^{(1)}(G)\ ]T_{14} = 0$ 122

$(b_{15})^{(1)}T_{14} - [(b'_{15})^{(1)} - (b''_{15})^{(1)}(G)\ ]T_{15} = 0$ 123

has a unique positive solution , which is an equilibrium solution for the system

$(a_{16})^{(2)}G_{17} - \left[(a'_{16})^{(2)} + (a''_{16})^{(2)}(T_{17})\right]G_{16} = 0$ 124

$(a_{17})^{(2)}G_{16} - \left[(a'_{17})^{(2)} + (a''_{17})^{(2)}(T_{17})\right]G_{17} = 0$ 125

$(a_{18})^{(2)}G_{17} - \left[(a'_{18})^{(2)} + (a''_{18})^{(2)}(T_{17})\right]G_{18} = 0$ 126

$(b_{16})^{(2)}T_{17} - [(b'_{16})^{(2)} - (b''_{16})^{(2)}(G_{19})\,]T_{16} = 0$ 127

$(b_{17})^{(2)}T_{16} - [(b'_{17})^{(2)} - (b''_{17})^{(2)}(G_{19})\,]T_{17} = 0$ 128

$(b_{18})^{(2)}T_{17} - [(b'_{18})^{(2)} - (b''_{18})^{(2)}(G_{19})\,]T_{18} = 0$ 129

has a unique positive solution , which is an equilibrium solution for THE GLOBAL SYSTEM:

Proof:

(a) Indeed the first two equations have a nontrivial solution G_{13}, G_{14} if

$F(T) = (a'_{13})^{(1)}(a'_{14})^{(1)} - (a_{13})^{(1)}(a_{14})^{(1)} + (a'_{13})^{(1)}(a''_{14})^{(1)}(T_{14}) + (a'_{14})^{(1)}(a''_{13})^{(1)}(T_{14}) + (a''_{13})^{(1)}(T_{14})(a''_{14})^{(1)}(T_{14}) = 0$

Proof: 130

(a) Indeed the first two equations have a nontrivial solution G_{16}, G_{17} if

$\mathrm{F}(T_{19}) = (a'_{16})^{(2)}(a'_{17})^{(2)} - (a_{16})^{(2)}(a_{17})^{(2)} + (a'_{16})^{(2)}(a''_{17})^{(2)}(T_{17}) + (a'_{17})^{(2)}(a''_{16})^{(2)}(T_{17}) + (a''_{16})^{(2)}(T_{17})(a''_{17})^{(2)}(T_{17}) = 0$

Definition and uniqueness of T_{14}^* :- 131

After hypothesis $f(0) < 0, f(\infty) > 0$ and the functions $(a''_i)^{(1)}(T_{14})$ are being increasing, it follows that there exists a unique T_{14}^* for which $f(T_{14}^*) = 0$. With this value , we obtain from the three first equations

$G_{13} = \frac{(a_{13})^{(1)}G_{14}}{[(a'_{13})^{(1)}+(a''_{13})^{(1)}(T_{14}^*)]}$, $G_{15} = \frac{(a_{15})^{(1)}G_{14}}{[(a'_{15})^{(1)}+(a''_{15})^{(1)}(T_{14}^*)]}$

Definition and uniqueness of T_{17}^* :- 132

After hypothesis $f(0) < 0, f(\infty) > 0$ and the functions $(a''_i)^{(2)}(T_{17})$ are increasing, it follows that there exists a unique T_{17}^* for which $f(\mathrm{T}_{17}^*) = 0$. With this value , we obtain from the three first equations

$G_{16} = \frac{(a_{16})^{(2)}\mathrm{G}_{17}}{[(a'_{16})^{(2)}+(a''_{16})^{(2)}(\mathrm{T}_{17}^*)]}$, $G_{18} = \frac{(a_{18})^{(2)}\mathrm{G}_{17}}{[(a'_{18})^{(2)}+(a''_{18})^{(2)}(\mathrm{T}_{17}^*)]}$ 133

(c) By the same argument, the equations OF THE SOLUTIONS OF THE GOVERNING EQUATIONS admit solutions G_{13}, G_{14} if

$\varphi(G) = (b'_{13})^{(1)}(b'_{14})^{(1)} - (b_{13})^{(1)}(b_{14})^{(1)} -$

$\big[(b'_{13})^{(1)}(b''_{14})^{(1)}(G) + (b'_{14})^{(1)}(b''_{13})^{(1)}(G)\big]+(b''_{13})^{(1)}(G)(b''_{14})^{(1)}(G) = 0$

Where in $G(G_{13}, G_{14}, G_{15}), G_{13}, G_{15}$ must be replaced by their values from 96. It is easy to see that φ is a decreasing function in G_{14} taking into account the hypothesis $\varphi(0) > 0\,, \varphi(\infty) < 0$ it follows that there exists a unique G_{14}^* such that $\varphi(G^*) = 0$

(d) By the same argument, the equations (SOLUTIONAL) OF THE UNIVESAL SYSTEM admit solutions G_{16}, G_{17} if 134

$\varphi(G_{19}) = (b'_{16})^{(2)}(b'_{17})^{(2)} - (b_{16})^{(2)}(b_{17})^{(2)} -$

$\left[(b'_{16})^{(2)}(b''_{17})^{(2)}(G_{19}) + (b'_{17})^{(2)}(b''_{16})^{(2)}(G_{19})\right] + (b''_{16})^{(2)}(G_{19})(b''_{17})^{(2)}(G_{19}) = 0$

Where in $(G_{19})(G_{16}, G_{17}, G_{18}), G_{16}, G_{18}$ must be replaced by their values from 96. It is easy to see that φ is a decreasing function in G_{17} taking into account the hypothesis $\varphi(0) > 0 , \varphi(\infty) < 0$ it follows that there exists a unique G^*_{14} such that $\varphi((G_{19})^*) = 0$

Finally we obtain the unique solution of 89 to 94 135

G^*_{14} given by $\varphi(G^*) = 0$, T^*_{14} given by $f(T^*_{14}) = 0$ and

$G^*_{13} = \frac{(a_{13})^{(1)}G^*_{14}}{[(a'_{13})^{(1)}+(a''_{13})^{(1)}(T^*_{14})]}$, $G^*_{15} = \frac{(a_{15})^{(1)}G^*_{14}}{[(a'_{15})^{(1)}+(a''_{15})^{(1)}(T^*_{14})]}$

$T^*_{13} = \frac{(b_{13})^{(1)}T^*_{14}}{[(b'_{13})^{(1)}-(b''_{13})^{(1)}(G^*)]}$, $T^*_{15} = \frac{(b_{15})^{(1)}T^*_{14}}{[(b'_{15})^{(1)}-(b''_{15})^{(1)}(G^*)]}$

Obviously, these values represent an equilibrium solution of GOVERNING EQUATIONS OF THE HOLISTIC SYSTEM:

Finally we obtain the unique solution of THE GOVERNING EQUATIONS

G^*_{17} given by $\varphi((G_{19})^*) = 0$, T^*_{17} given by $f(T^*_{17}) = 0$ and 136

$G^*_{16} = \frac{(a_{16})^{(2)}G^*_{17}}{[(a'_{16})^{(2)}+(a''_{16})^{(2)}(T^*_{17})]}$, $G^*_{18} = \frac{(a_{18})^{(2)}G^*_{17}}{[(a'_{18})^{(2)}+(a''_{18})^{(2)}(T^*_{17})]}$

$T^*_{16} = \frac{(b_{16})^{(2)}T^*_{17}}{[(b'_{16})^{(2)}-(b''_{16})^{(2)}((G_{19})^*)]}$, $T^*_{18} = \frac{(b_{18})^{(2)}T^*_{17}}{[(b'_{18})^{(2)}-(b''_{18})^{(2)}((G_{19})^*)]}$

Obviously, these values represent an equilibrium solution of THE GOVERNING EQUATIONS OF THE GLOBAL SYSTEM"P" VIS A VIS "NP" CONCATENATED WITH "NPC" VIS A VIS NP(HARD)

ASYMPTOTIC STABILITY ANALYSIS OF THE GLOBAL SYSTEM"P" VIS A VIS "NP" CONCATENATED WITH "NPC" VIS A VIS NP(HARD) 137

Theorem 4: If the conditions of the previous theorem are satisfied and if the functions $(a''_i)^{(1)}$ *and* $(b''_i)^{(1)}$ Belong to $C^{(1)}(\mathbb{R}_+)$ then the above equilibrium point is asymptotically stable.

Proof:_Denote

Definition of $\mathbb{G}_i, \mathbb{T}_i$:-

$G_i = G^*_i + \mathbb{G}_i$, $T_i = T^*_i + \mathbb{T}_i$

$\frac{\partial (a''_{14})^{(1)}}{\partial T_{14}}(T^*_{14}) = (q_{14})^{(1)}$, $\frac{\partial (b''_i)^{(1)}}{\partial G_j}(G^*) = s_{ij}$

Then taking into account equations SOLUTIONAL EQUATIONS OF THE UNIVERSAL EQUATIONS DILATED IN THE FOREGOING and neglecting the terms of power 2, we obtain

$\frac{d\mathbb{G}_{13}}{dt} = -\left((a'_{13})^{(1)} + (p_{13})^{(1)}\right)\mathbb{G}_{13} + (a_{13})^{(1)}\mathbb{G}_{14} - (q_{13})^{(1)}G^*_{13}\mathbb{T}_{14}$ 13

8

$$\frac{d\mathbb{G}_{14}}{dt} = -\left((a'_{14})^{(1)} + (p_{14})^{(1)}\right)\mathbb{G}_{14} + (a_{14})^{(1)}\mathbb{G}_{13} - (q_{14})^{(1)}G^*_{14}\mathbb{T}_{14}$$ 139

$$\frac{d\mathbb{G}_{15}}{dt} = -\left((a'_{15})^{(1)} + (p_{15})^{(1)}\right)\mathbb{G}_{15} + (a_{15})^{(1)}\mathbb{G}_{14} - (q_{15})^{(1)}G^*_{15}\mathbb{T}_{14}$$ 140

$$\frac{d\mathbb{T}_{13}}{dt} = -\left((b'_{13})^{(1)} - (r_{13})^{(1)}\right)\mathbb{T}_{13} + (b_{13})^{(1)}\mathbb{T}_{14} + \sum_{j=13}^{15}\left(s_{(13)(j)}T^*_{13}\mathbb{G}_j\right)$$ 141

$$\frac{d\mathbb{T}_{14}}{dt} = -\left((b'_{14})^{(1)} - (r_{14})^{(1)}\right)\mathbb{T}_{14} + (b_{14})^{(1)}\mathbb{T}_{13} + \sum_{j=13}^{15}\left(s_{(14)(j)}T^*_{14}\mathbb{G}_j\right)$$ 142

$$\frac{d\mathbb{T}_{15}}{dt} = -\left((b'_{15})^{(1)} - (r_{15})^{(1)}\right)\mathbb{T}_{15} + (b_{15})^{(1)}\mathbb{T}_{14} + \sum_{j=13}^{15}\left(s_{(15)(j)}T^*_{15}\mathbb{G}_j\right)$$ 143

If the conditions of the previous theorem are satisfied and if the functions $(a''_i)^{(2)}$ and $(b''_i)^{(2)}$ Belong to $C^{(2)}(\mathbb{R}_+)$ then the above equilibrium point is asymptotically stable

<u>Definition of</u> $\mathbb{G}_i, \mathbb{T}_i$:-

$$G_i = G^*_i + \mathbb{G}_i \qquad , T_i = T^*_i + \mathbb{T}_i$$ 144

$$\frac{\partial (a''_{17})^{(2)}}{\partial T_{17}}(T^*_{17}) = (q_{17})^{(2)} \quad , \frac{\partial (b''_i)^{(2)}}{\partial G_j}\left((G_{19})^*\right) = s_{ij}$$

taking into account SOLUTIONAL equations and neglecting the terms of power 2, we obtain

$$\frac{d\mathbb{G}_{16}}{dt} = -\left((a'_{16})^{(2)} + (p_{16})^{(2)}\right)\mathbb{G}_{16} + (a_{16})^{(2)}\mathbb{G}_{17} - (q_{16})^{(2)}G^*_{16}\mathbb{T}_{17}$$ 145

$$\frac{d\mathbb{G}_{17}}{dt} = -\left((a'_{17})^{(2)} + (p_{17})^{(2)}\right)\mathbb{G}_{17} + (a_{17})^{(2)}\mathbb{G}_{16} - (q_{17})^{(2)}G^*_{17}\mathbb{T}_{17}$$ 146

$$\frac{d\mathbb{G}_{18}}{dt} = -\left((a'_{18})^{(2)} + (p_{18})^{(2)}\right)\mathbb{G}_{18} + (a_{18})^{(2)}\mathbb{G}_{17} - (q_{18})^{(2)}G^*_{18}\mathbb{T}_{17}$$ 147

$$\frac{d\mathbb{T}_{16}}{dt} = -\left((b'_{16})^{(2)} - (r_{16})^{(2)}\right)\mathbb{T}_{16} + (b_{16})^{(2)}\mathbb{T}_{17} + \sum_{j=16}^{18}\left(s_{(16)(j)}T^*_{16}\mathbb{G}_j\right)$$ 148

$$\frac{d\mathbb{T}_{17}}{dt} = -\left((b'_{17})^{(2)} - (r_{17})^{(2)}\right)\mathbb{T}_{17} + (b_{17})^{(2)}\mathbb{T}_{16} + \sum_{j=16}^{18}\left(s_{(17)(j)}T^*_{17}\mathbb{G}_j\right)$$ 149

$$\frac{d\mathbb{T}_{18}}{dt} = -\left((b'_{18})^{(2)} - (r_{18})^{(2)}\right)\mathbb{T}_{18} + (b_{18})^{(2)}\mathbb{T}_{17} + \sum_{j=16}^{18}\left(s_{(18)(j)}T^*_{18}\mathbb{G}_j\right)$$ 150

The characteristic equation of this system is 151

$((\lambda)^{(1)} + (b_{15}')^{(1)} - (r_{15})^{(1)})\{((\lambda)^{(1)} + (a_{15}')^{(1)} + (p_{15})^{(1)})$

$\left[\left(((\lambda)^{(1)} + (a_{13}')^{(1)} + (p_{13})^{(1)})(q_{14})^{(1)}G_{14}^{*} + (a_{14})^{(1)}(q_{13})^{(1)}G_{13}^{*}\right)\right]$

$\left(((\lambda)^{(1)} + (b_{13}')^{(1)} - (r_{13})^{(1)})s_{(14),(14)}T_{14}^{*} + (b_{14})^{(1)}s_{(13),(14)}T_{14}^{*}\right)$

$+\left(((\lambda)^{(1)} + (a_{14}')^{(1)} + (p_{14})^{(1)})(q_{13})^{(1)}G_{13}^{*} + (a_{13})^{(1)}(q_{14})^{(1)}G_{14}^{*}\right)$

$\left(((\lambda)^{(1)} + (b_{13}')^{(1)} - (r_{13})^{(1)})s_{(14),(13)}T_{14}^{*} + (b_{14})^{(1)}s_{(13),(13)}T_{13}^{*}\right)$

$\left(((\lambda)^{(1)})^{2} + \left((a_{13}')^{(1)} + (a_{14}')^{(1)} + (p_{13})^{(1)} + (p_{14})^{(1)}\right)(\lambda)^{(1)}\right)$

$\left(((\lambda)^{(1)})^{2} + \left((b_{13}')^{(1)} + (b_{14}')^{(1)} - (r_{13})^{(1)} + (r_{14})^{(1)}\right)(\lambda)^{(1)}\right)$

$+\left(((\lambda)^{(1)})^{2} + \left((a_{13}')^{(1)} + (a_{14}')^{(1)} + (p_{13})^{(1)} + (p_{14})^{(1)}\right)(\lambda)^{(1)}\right)(q_{15})^{(1)}G_{15}$

$+((\lambda)^{(1)} + (a_{13}')^{(1)} + (p_{13})^{(1)})\left((a_{15})^{(1)}(q_{14})^{(1)}G_{14}^{*} + (a_{14})^{(1)}(a_{15})^{(1)}(q_{13})^{(1)}G_{13}^{*}\right)$

$\left(((\lambda)^{(1)} + (b_{13}')^{(1)} - (r_{13})^{(1)})s_{(14),(15)}T_{14}^{*} + (b_{14})^{(1)}s_{(13),(15)}T_{13}^{*}\right)\} = 0$ 152

$+$

$((\lambda)^{(2)} + (b_{18}')^{(2)} - (r_{18})^{(2)})\{((\lambda)^{(2)} + (a_{18}')^{(2)} + (p_{18})^{(2)})$

$\left[\left(((\lambda)^{(2)} + (a_{16}')^{(2)} + (p_{16})^{(2)})(q_{17})^{(2)}G_{17}^{*} + (a_{17})^{(2)}(q_{16})^{(2)}G_{16}^{*}\right)\right]$

$\left(((\lambda)^{(2)} + (b_{16}')^{(2)} - (r_{16})^{(2)})s_{(17),(17)}T_{17}^{*} + (b_{17})^{(2)}s_{(16),(17)}T_{17}^{*}\right)$

$+\left(((\lambda)^{(2)} + (a_{17}')^{(2)} + (p_{17})^{(2)})(q_{16})^{(2)}G_{16}^{*} + (a_{16})^{(2)}(q_{17})^{(2)}G_{17}^{*}\right)$ 153

$\left(((\lambda)^{(2)} + (b_{16}')^{(2)} - (r_{16})^{(2)})s_{(17),(16)}T_{17}^{*} + (b_{17})^{(2)}s_{(16),(16)}T_{16}^{*}\right)$

$\left(((\lambda)^{(2)})^{2} + \left((a_{16}')^{(2)} + (a_{17}')^{(2)} + (p_{16})^{(2)} + (p_{17})^{(2)}\right)(\lambda)^{(2)}\right)$

$\left(((\lambda)^{(2)})^{2} + \left((b_{16}')^{(2)} + (b_{17}')^{(2)} - (r_{16})^{(2)} + (r_{17})^{(2)}\right)(\lambda)^{(2)}\right)$

$+\left(((\lambda)^{(2)})^{2} + \left((a_{16}')^{(2)} + (a_{17}')^{(2)} + (p_{16})^{(2)} + (p_{17})^{(2)}\right)(\lambda)^{(2)}\right)(q_{18})^{(2)}G_{18}$

$+((\lambda)^{(2)} + (a_{16}')^{(2)} + (p_{16})^{(2)})\left((a_{18})^{(2)}(q_{17})^{(2)}G_{17}^{*} + (a_{17})^{(2)}(a_{18})^{(2)}(q_{16})^{(2)}G_{16}^{*}\right)$

$\left(((\lambda)^{(2)} + (b_{16}')^{(2)} - (r_{16})^{(2)})s_{(17),(18)}T_{17}^{*} + (b_{17})^{(2)}s_{(16),(18)}T_{16}^{*}\right)\} = 0$

And as one sees, all the coefficients are positive. It follows that all the roots have negative real part, and this proves the theorem.

Acknowledgments:

The introduction is a collection of information from various articles, Books, News Paper reports,

Home Pages Of authors, Journal Reviews, the internet including Wikipedia. We acknowledge all authors who have contributed to the same. In the eventuality of the fact that there has been any act of omission on the part of the authors, We regret with great deal of compunction, contrition, and remorse. As Newton said, it is only because erudite and eminent people allowed one to piggy ride on their backs; probably an attempt has been made to look slightly further. Once again, it is stated that the references are only illustrative and not comprehensive

REFERENCES

1. Dr K N Prasanna Kumar, Prof B S Kiranagi, Prof C S Bagewadi - MEASUREMENT DISTURBS EXPLANATION OF QUANTUM MECHANICAL STATES-A HIDDEN VARIABLE THEORY - published at: "*International Journal of Scientific and Research Publications, Volume 2, Issue 5, May 2012 Edition*".

2. DR K N PRASANNA KUMAR, PROF B S KIRANAGI and PROF C S BAGEWADI -CLASSIC 2 FLAVOUR COLOR SUPERCONDUCTIVITY AND ORDINARY NUCLEAR MATTER-A NEW PARADIGM STATEMENT - published at: "*International Journal of Scientific and Research Publications, Volume 2, Issue 5, May 2012 Edition*".

3. A HAIMOVICI: “On the growth of a two species ecological system divided on age groups”. Tensor, Vol 37 (1982),Commemoration volume dedicated to Professor Akitsugu Kawaguchi on his 80th birthday

4. FRTJOF CAPRA: “The web of life” Flamingo, Harper Collins See "Dissipative structures” pages 172-188

5. HEYLIGHEN F. (2001): "The Science of Self-organization and Adaptivity", in L. D. Kiel, (ed) . Knowledge Management, Organizational Intelligence and Learning, and Complexity, in: The Encyclopedia of Life Support Systems ((EOLSS), (Eolss Publishers, Oxford) [http://www.eolss.net

6. MATSUI, T, H. Masunaga, S. M. Kreidenweis, R. A. Pielke Sr., W.-K. Tao, M. Chin, and Y. J Kaufman (2006), “Satellite-based assessment of marine low cloud variability associated with aerosol, atmospheric stability, and the diurnal cycle”, J. Geophys. Res., 111, D17204, doi:10.1029/2005JD006097

7. STEVENS, B, G. Feingold, W.R. Cotton and R.L. Walko, “Elements of the microphysical structure of numerically simulated nonprecipitating stratocumulus” J. Atmos. Sci., 53, 980-1006

8. FEINGOLD, G, Koren, I; Wang, HL; Xue, HW; Brewer, WA (2010), “Precipitation-generated oscillations in open cellular cloud fields” *Nature*, *466* (7308) 849-852, doi: 10.1038/nature09314, Published 12-Aug 2010

9. R WOOD “The rate of loss of cloud droplets by coalescence in warm clouds” J.Geophys. Res., 111, doi: 10.1029/2006JD007553, 2006

10. H. RUND, “The Differential Geometry of Finsler Spaces”, Grund. Math. Wiss. Springer-Verlag, Berlin, 1959

11. A. Dold, “Lectures on Algebraic Topology”, 1972, Springer-Verlag

12. S LEVIN “Some Mathematical questions in Biology vii ,Lectures on Mathematics in life sciences, vol 8” The American Mathematical society, Providence , Rhode island 1976

First Author: [1]**Mr. K. N.Prasanna Kumar** has three doctorates one each in Mathematics, Economics, Political Science. Thesis was based on Mathematical Modeling. He was recently awarded D.litt. for his work on ‘Mathematical Models in Political Science’--- Department of studies in Mathematics, Kuvempu University, Shimoga, Karnataka, India Corresponding **Author:drknpkumar@gmail.com**

Second Author: [2]**Prof. B.S Kiranagi** is the Former Chairman of the Department of Studies in Mathematics, Manasa Gangotri and present Professor Emeritus of UGC in the Department. Professor Kiranagi has guided over 25 students and he has received many encomiums and laurels for his contribution to Co homology Groups and Mathematical Sciences. Known for his prolific writing, and one of the senior most Professors of the country, he has over 150 publications to his credit. A prolific writer and a prodigious thinker, he has to his credit several books on Lie Groups, Co Homology Groups, and other mathematical application topics, and excellent publication history.-- UGC Emeritus Professor (Department of studies in Mathematics), Manasagangotri, University of Mysore, Karnataka, India

Third Author: [3]**Prof. C.S. Bagewadi** is the present Chairman of Department of Mathematics and Department of Studies in Computer Science and has guided over 25 students. He has published articles in both national and international journals. Professor Bagewadi specializes in Differential Geometry and its wide-ranging ramifications. He has to his credit more than 159 research papers. Several Books on Differential Geometry, Differential Equations are coauthored by him--- Chairman, Department of studies in Mathematics and Computer science, Jnanasahyadri Kuvempu University, Shankarghatta, Shimoga district, Karnataka, India

==
====