# Rethinking Embedded System Design

**Subhendu Das** *subhendu11das@gmail.com*
*CCSI*
*West Hills, 91307, USA*

**Abstract**

Embedded engineering is designed using objects of nature and it also interacts with nature. Therefore it is forced to obey the laws of nature. Nature does not make any assumptions. But all our mathematical and scientific theories do. Therefore these theories cannot be valid for embedded engineering applications. In this paper we present four new laws of nature that all embedded systems follow. These laws are (1) Boundedness (2) Finite time (3) Simultaneity and (4) Complexity. During the last fifty years embedded analog and digital engineering have evolved and changed significantly. However our mathematical and scientific theories remained in the original state. We select several theories commonly used in embedded engineering and show that none of them satisfy these laws. As a result, when we implement these theories in our embedded software, we are forced to add so many patches and kludges to make the engineering work, that our systems become very unreliable.

**Keywords**: Software, Embedded, Mathematics, Science, Engineering, Kalman, Education.

## 1. INTRODUCTIONS

In this paper our focus is on the problem of embedded system software and its failure to provide reliability and stability [1]. The embedded software has two categories of problems. The first problem is related to its own software design, implementation, and test issues. The second problem is related to the algorithm and theories that the software uses in its design. In this paper we are concerned with this second problem.

We believe that if we understand the root cause of the problem then we will be able to automatically find the solution. Unlike internet or PC type software, embedded system is part of nature. Therefore it must follow the laws of nature. We discuss four natural laws that all embedded systems follow: (a) boundedness law of all physical variables (b) finite time law of all activities, (c) simultaneity law of natural phenomena, and (d) the complexity law of nature. If we examine any embedded system software and hardware, we will find that all four laws are implemented very carefully. On the other hand if we examine the mathematical and scientific theories with similar care we will find none of them obey the above four laws. These inconsistencies are the root cause of the software unreliability issues of the second problem mentioned above.

It is not that we did not know about these laws. They existed in nature for eternity. They slowly started emerging as our engineering became more and more complex. We started integrating many subsystems together; and we configured them to communicate with each other, to create more realistic systems. Eventually we implemented real time multitasking software, which highlighted our focus on finite time repeating concepts and the simultaneity law of nature. The technology advanced so rapidly and implemented things in such details, thus increasing the complexity, that other branches of mathematics and science did not get opportunities to realize it. As a result we see a crisis in embedded system software and in our theories.

It must be recognized that our engineering requirements are significantly more complex [2] than what they were during the time of Newton. As an example, we may say that the first missile must make a hole in the building and the second missile must go through that hole. We have used this kind of technology in our recent battle fields of Iraq during 2003. This kind of precise requirements

cannot be implemented using the theories that still inherit the characteristics of the simpler days of Newton. The problems are in the details; math and science do not go at that level of details, only engineering does. Note that this is not about approximations; this is on gross violations of the laws of nature. Our embedded technology has significantly advanced during the last fifty years, but our math and science did not keep pace with it.

Using standard examples from mathematics and science we show that they cannot work in embedded engineering because they violate the new laws presented. We use (a) all three Newton's laws, (b) Laplace and Fourier transforms, and (c) Kalman filtering to present the concepts of this paper.

The objective in this paper is only to discuss the problems, and highlight their root causes. The problem itself is very complex and big. It should be realized that the details must be avoided to present the comprehensive nature of the problem in a paper of this size. In reality all solutions are embedded in the detailed understanding of the problems. It is possible to create new architectures, based on Kalman's philosophy of using only measurements, and thus help to produce reliable embedded software. We present, however, an alternative approach little bit, near the end. The paper may appear like a philosophical presentation to people in the areas of mathematics, science, analysis, and simulation etc., but for all experienced hands-on engineers it will be quite realistic, obvious, and normal.

## 2: DEFINITIONS

Before we talk about math, science, engineering, their theories, assumptions, and compare them with new laws mentioned; it may be necessary to define the terminologies. Nature has only two kinds of things; some objects (living and non-living) and some actions. Actions are like forces of nature and have some energy associated with them. In some sense actions are characteristics of objects also. For example light energy is a characteristic of sun; similarly wind force is a characteristic of earth.

**Definition of Laws of Nature**

The laws of nature are the universal characteristics of the objects of nature. They are physical. They exist independent of human experiences and assumptions.

Everything that we see around us is engineering. The cars, airplanes, roads, buildings are all products of engineering. A product is a physical hardware that we can touch. Our modern engineering products are very sophisticated and satisfy complex requirements.

**Definition of Engineering**

It is a process that is required to create an useful product.

Thus engineering is not the textbooks on engineering subjects, like mechanical, electrical, etc. All products use natural components, and therefore they also obey natural laws. Thus we can define science in the following way:

**Definition of Science**

It is a collection of manmade theories that tries to explain the laws of nature.

Consider an example to clarify the distinction between science and engineering. If we place a magnetic needle under a wire, and pass current through the wire, then the magnet will be deflected. We call this an engineering experiment. It is a product that we can see, touch, and learn about it; and it does something useful also. The process used to demonstrate this needle movement is engineering. The science part says that the magnet has a field called magnetic field, the electricity creates a field called electric field (or may be a magnetic field); these two fields interact and create a force that deflects the magnet. The mathematics is a symbolic language. Its main purpose is to justify the scientific theories.

**Definition of Mathematics**
It is a symbolic language, used to describe expressions of natural language.
The theory is always a set of conclusions or a set of rules. But it also says that these rules or results will hold only under certain assumptions. These assumptions are thus a part of the theory.

**Definition of Theory**
A Theory is (a) a collection of assumptions and (b) a collection of conclusions that only hold under the assumptions.

**Example of a Theory**
Newton's First law: (a) In the absence of any interaction with something else (b) An object at rest will remain at rest (c) An object in motion will continue in motion at constant velocity, that is, in constant speed in a straight line

The item (a) in the above law is the assumption. The items (b) and (c) are the conclusions. The last two items will be valid only when the first item (a) is valid. A theory has two parts, if any one of the two parts fails then that theory will be invalid and we will say that the theory does not work in engineering or simply does not work.

**Definition of Invalidity**
A Theory is invalid if (a) Its assumptions cannot be tested or implemented or (b) Its conclusions cannot be verified by any experiment

## 3: THE LAWS OF NATURE

In this section we present the following new laws of nature: (a) boundedness law (b) finite time law (c) simultaneity law and (d) the complexity law. We describe them in details in the subsections below.

### 3.1 Boundedness Law

Let x denote any engineering variable, like voltage, current, water pressure, etc. In engineering x always has a lowest and a highest possible value. Or in other words they cannot take any arbitrary value from minus infinity to plus infinity. We call this feature of a variable as the boundedness law of nature. We will also call it the nonlinearity or saturation law of engineering. Using mathematical notations we can express this law in the following way:

$$L \leq x \leq U \tag{1}$$

Here L can be positive, zero, or any negative number but cannot be minus infinity. Similarly U can be negative, zero, or any positive number, but cannot be positive infinity. However U must be greater than L.

Figure 1 describes graphically the logic to implement (1). The horizontal axis is the input axis for the variable x. Along the horizontal axis, x can take any value from minus infinity to plus infinity. We show the boundedness of x in the vertical or output axis of the graph. The graph shows when x is between a and b, whatever way x changes, similar changes happen in the output axis. However, when x goes beyond b you can see that on the output axis it stays fixed at U. That is the output is limited to U. The same is true in the lower direction of x, and is limited by L. The box in Figure 1 represents a non-linear system.

Wherever there is an engineering variable in a system, an embedded engineer adds this box at that location of the system, to protect the components there, otherwise the system will fail or burnout. The box is also known as limiter. This box is implemented in analog hardware circuits in the form automatic gain control. It is also implemented in both digital hardware and embedded microprocessor based software. All hands-on experienced engineers will automatically place these boxes in their design. This is a very common embedded engineering practice.

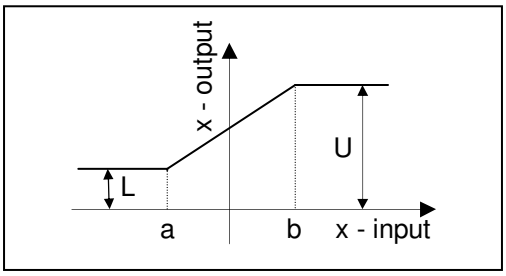


**FIGURE 1:** Saturation Non-Linearity

Since the box is a nonlinear box representing the natural law, and implemented by design, all engineering systems are nonlinear systems by nature. Or in other words there are no linear systems in engineering. Note that linear systems are not approximations to this kind of boundedness law. We explain later that local linearized solutions do not satisfy modern complex engineering requirements. They will violate the complexity law mentioned below.

The examples of such systems with boundedness law are abundant in engineering. All we have to look for is to see the schematic of any embedded hardware and the source code of real time embedded software or firmware. It is not that engineering has completely ignored it, theory has been developed, as an example see [3]. But it is still not practical for the complex problems of our time. However mathematics and science have largely ignored the boundedness law, as we show later with examples. Observe that this is not a philosophical issue, it is a real engineering problem, and creates conflict with existing theories that we use.

### 3.2 Finite Time Law

These days most of the complex engineering systems are controlled by one or more digital microprocessors and software. All activities that these systems perform are done in small interval of time duration, of the order of several micro or milliseconds. And such activities are repeated continuously [4].

Consider the example of a robotic arm, picking up an item from one place and dropping it in another place and repeating the process in, say, less than a second of time. Similarly a digital communication receiver system, like GPS, receives an electrical signal of microsecond duration, for example, representing the data, extracts the data from the signal, sends it to the output, and then goes back to repeat the process.

Our software runs under real time multitasking operating systems which are also nothing but finite state machines. A finite state machine is a collection of finite number of activities of finite durations, repeated asynchronously and/or synchronously based on the external as well as internal events. Every time a task returns, it finds a different environment. The previous tasks have operated on the system and created a new environment. Thus the same finite duration task or activity is always performed on different signal and under different environment.

Although our systems run continuously, like GPS transmitters and receivers, traffic light systems at street corners, but if you look at the internals you will always find that the building blocks are based on finite duration processes.

It is quite surprising that the embedded system evolved to perform things as continuous repetition of finite time activities. If we observe carefully we will see that the nature is also composed of finite time activities. Everything in nature has a birth process, maturity process, and death process. Each one of the processes is also a finite duration process. We also see that earth rotates over finite time around its axis and around the sun. Thus repeating finite time process is a law of nature. Yet, most of the mathematical theories that we commonly use, assume infinite

time, as we show later with examples. We also show that if we replace the infinite time by finite number, however large, the theory completely fails. Thus again, these are not approximations of large numbers, these are collapse of theories, as explained later.

### 3.3 Simultaneity Law

A very important characteristic of nature is its simultaneity. Everything in nature occurs simultaneously and interactively. All humans are interacting constantly, simultaneously, and all over the world and for all time. So is true with all physical objects. We are never isolated. A company on precision weight measurement system [5] uses the moon's gravity effect, as it travels over earth, to precisely measure the weight of a mass on earth. Thus simultaneity is global and not just local. This company's products show how complex and sophisticated our modern requirements are. Before we even realize, everything in engineering will be simultaneously integrated together just like our natural world is. But our math and science are not yet ready for it. Most of the theories that we use now are more than hundred years old. Requirements, concepts, and philosophies of those simpler days are deeply embedded in those theories.

The real time operating system (RTOS) also implements this simultaneity law in engineering. It is a multitasking system that interacts with interrupts from external and internal sources. Basically RTOS is a finite state machine, running on finite time intervals, and is designed to simultaneously accept the changes in the environment. Many of our embedded systems interact with external computers via serial interfaces. In many cases these interfaces bring user commands also. These interfaces are constantly monitored by several tasks to reconfigure the system according to the changes in the environment. Thus simultaneity is built into all embedded systems.

Clearly, RTOS is beyond the scope of mathematics and science, but it is an integral part of modern engineering. Anytime a task switches from one to another, it finds the environment completely changed. When the task was in sleep mode, the simultaneity law worked and changed that environment. But our present theories rely on the continuity of states from one task to another, but that does not hold under RTOS and all other laws discussed here. We show later that most of our theories do not have means to accommodate all the laws mentioned,

### 3.4 Complexity Law

All natural systems are immensely complex and indescribable. To illustrate, consider the Grand Canyon. If we ask the best author of the world to describe the Grand Canyon in written language; you will find that the description will be of no match with your experience and feeling when you personally see the Grand Canyon. This written document is a model of the Grand Canyon. Thus nature is beyond description by our language and therefore cannot be modeled by a symbolic language like mathematics. The Grand Canyon is a static example of complexity. The dynamic complexity of nature is even more severe than Grand Canyon. Here is one more example to convince the readers about the complexity law. Watch the 3D simulation of the human brain in operation from the discovery channel [6] to comprehend the nature. This is only a simulation; the real thing is actually lot more complex.

Nature has evolved over billions of years. As a result everything is very complex in nature. From a very small thing like an atom to a very large system like a galaxy are all very complex. We should recognize that not only the components are complex; the laws that govern them are also equally complex. Our engineering uses these complex components from nature and implements these laws of nature to make products that are supposed to satisfy very complex requirements. Thus embedded engineering is as complex as nature.

Imagine what is inside a microprocessor. It has billions of electronic components inside it. The processor has hundreds or thousands of 32-bit registers, each bit must be carefully programmed to make it work according to desired performances and requirements. These registers exchange and process information with nature via analog to digital and digital to analog converters at nanoseconds and microseconds speed. The numbers inside the microprocessor also change continuously because of variations in nature, like changes in temperature, drift of component

parameters etc. The nature looks completely different at that level of speed and 32-bit details. Our mathematics, which was developed hundreds of years back, cannot comprehend such complexities.

Even today many embedded software do not use floating point processors. The integer processors require scaling of variables. Scaling is a nonlinear process to keep variables within bounds, essentially implementing the boundedness law. This statics scaling process can never work, in real time and under the circumstances governed by the laws mentioned. Texas Instrument, which manufactured such a processor finally decided to make a floating point version to accommodate requirements of embedded engineering. Matlab simulation software created dynamic scaling every time it scaled a variable in its simulation to give correct results. This approach, although correct, is clearly not feasible in real time engineering.

All the previous three laws, boundedness, finite time, and simultaneity, are all working together in nature and therefore also in embedded engineering. This togetherness adds another dimension to the complexity of nature. We call this complex nature as the global space time (GST) environment and it is tightly integrated with all embedded systems. We should also point out that simulation environment cannot be created to test out such a complex and simultaneous environment for a real time embedded engineering system.

## 4. VIOLATING THE LAWS

In embedded engineering software, we use many mathematical and scientific algorithms. None of these theories obey the new laws mentioned in the previous section. In the following subsections we take many well known theories that are often used in embedded systems and show their incompatibility with these laws of nature. All these theories make simplifying assumptions, but nature does and cannot make any assumptions. Note that we are not trying to find faults in these theories; they work perfectly according to their definitions and assumptions. All we are saying is that they were invented hundreds of years back and not valid any more for our modern engineering. Engineering has advanced significantly during the last fifty years but our theories did not make similar progress.

When we implement these theories they fail to work. But the software and test engineers know how to fix them. They are forced to add many adhoc patches and kludges to make the engineering work. That is why we say math and science do not work, but engineering does. However, as a result of these patches the embedded system remains very unreliable and unpredictable.

The two worlds, designers and testers, are isolated and do not know the tools and languages of each other and cannot communicate. We see an interesting parallel highlighted [7] by the Nobel Laureate in economics Wassily Leontief – “How long will researchers working in adjoining fields abstain from expressing serious concern about the splendid isolation in which economics now finds itself?” System engineers do not seem to recognize the inadequacy of our mathematical and scientific theories, which are in complete isolation with embedded engineering.

### 4.1 Linear Theories

The above boundedness law shows that all engineering systems are nonlinear by design requirements. Therefore all mathematical and scientific theories that are based on the assumption of linearity are no good for any engineering system. As an example, linear control system theories, based on linear Laplace transform theory cannot work. Their applications to engineering problems are theoretically incorrect. Violating the boundedness law is not an approximation; it is a violation of fundamentals. It will lead to wrong results, instability, and system crashes. According to our definition of invalidity, Kalman filter, Laplace transform, and linear control theory cannot work in embedded systems because they violate the assumption of boundedness.

Most of the engineering systems have multiple modes of operations, like transient mode, steady state mode, low voltage mode, high voltage mode etc. In many cases, abnormal situations happen and cannot be avoided. This boundedness law protects the systems from failures. In almost all cases there is no theory that considers such assumptions and possibilities in their theoretical proof, mostly because they are all linear theories and therefore ignore this boundedness law.

Because of the boundedness law, in a typical engineering system there can be more than ten such nonlinear boxes. In many large systems there can be hundreds of such boxes. In most cases we have to show that the system is working properly at the boundary points of the nonlinearity. In many applications it will be a requirement to go to the boundary points L and U of Figure 1, and maintain the operating conditions at these limits. For example, the motor must run at the maximum speed, the voltage used must be of full value, the angular position must reach the limiting position etc. to achieve the desired performances.

It will be rarely required to operate the system in the linear region of the saturation nonlinearity. Moreover, when there are more than one or two or ten such nonlinear boxes in a system, it will be almost impossible to keep all variables simultaneously in the linear region of the saturating boxes, and still maintain the optimal performance. This will happen because the nonlinear equations that define the operation of the system, such as (3) below, will not necessarily create equilibrium positions in the linear region at all operating conditions.

Besides the boundedness law, almost all systems are also nonlinear. These systems are modeled using nonlinear equations. For example expression (2) is a linear model but (3) is not.

$$\frac{dx}{dt} = ax \tag{2}$$

$$\frac{dx}{dt} = bx + cx^2 \tag{3}$$

In an isolated environment, as discussed later, (3) can be linearized to (2) and linear theories can be used as approximations. However under boundedness law both (2) and (3) will fail to work. Moreover, there are no isolated systems or environments in nature or engineering.

### 4.2 Laplace and Fourier Transforms

Under the operational environment of the finite time law any scientific or mathematical theory that is based on infinite time assumption will be inappropriate for embedded engineering systems. According to our definition they will be invalid, because they violate the finite time assumptions. Consider for example the very well known Laplace transform theory defined by (4):

$$F(s) = \int_0^\infty e^{-st} f(t)dt \tag{4}$$

In the expression (4) the variable t is usually considered as time. It shows that the time must be valid from zero to infinity. Thus the function F(s) on the left hand side, called infinite Laplace transform (ILT), will be valid only for infinite time system. Thus a major theory, Laplace Transform, of mathematics is invalid for embedded engineering applications. It is also well known that (4) is based on the assumption of linearity. And therefore it will also not be applicable for any engineering systems under the boundedness law.

Let us consider an example with finite time T to bring out the fact [8] that the Laplace transform is based on infinite time assumptions, that is, it cannot be used for finite duration signals. The finite duration step function *f(t)* is defined by

$$f(t) = \begin{cases} 1 & 0 \le t \le T \\ 0 & otherwise \end{cases} \tag{5}$$

Using the definition (4) we get the expression for the finite Laplace transform (FLT):

$$\mathcal{L}_T(1) = \int_0^T e^{-st}.1.dt$$

$$= \frac{1}{s} - \frac{1}{s}e^{-sT} \quad (6)$$

$$= \frac{1 - e^{-sT}}{s} \quad (7)$$

We can see from (6) that the FLT has the standard ILT term $\frac{1}{s}$ and another expression involving $e^{-sT}$. The second term is zero only when T is infinity. Thus if we use only the first part of (6), the ILT part, then we will implicitly assume infinite time situation for our finite time problems and the Laplace model will not be correct for real engineering problems.

Observe from expression (7) that the FLT does not have any poles at the origin, but the ILT has. At s=0 the expression (7) takes 0/0 form. Thus by using the first part of (6) for finite time engineering problems we artificially inject poles in the models. The entire Laplace Transform theory must be revised and rewritten for applications in finite time engineering [9]. For embedded engineering applications ILT is not a correct tool and its use will make the software unreliable.

Similarly, data analysis that we have done using infinite time Fourier theory could be wrong also [4]. If you analyze this way then you will find that all mathematical theories will be invalid under all four laws. It has been shown that by switching from infinite time to finite time Nyquist Sampling theorem changes [10]. This new sampling theorem can change the design of embedded systems significantly, and may even give a new direction for embedded design.

### 4.3 Newton's First Law

The assumption behind the first law is – "In the absence of any interaction with something else". Is there a place on earth where there is no interaction with something else? The answer is no. Everything in nature is tied together by the simultaneity law. Embedded engineering is embracing that law more and more in its implementations. Therefore the first law will become invalid, by our definition, if we use in embedded engineering.

You cannot place a ball above the surface of earth and leave it alone there, because earth is interacting with the ball and it will make it fall. Thus in near earth all objects are interacting with earth's gravitational force. If you put the ball in deep space, then it will face the gravitational attractions of sun, earth, moon, and all other planets. Their resultant force will not be zero. Thus there is no place in the universe where there is no interaction with something else.

If you leave the ball in the deep space, it will immediately start moving in a curved path. The path will be curved because the objects in our universe are constantly moving. The total force on the ball will be changing all the time, both in magnitude and in direction; therefore it cannot go in a straight line. Thus we see that the conclusions of the law cannot be true. The conclusion is false because the assumption is wrong. Newton [1642-1727] discovered these laws almost three hundred years back. We should not expect that they will work now for our modern engineering.

The following statements can be found in the textbook [11, p. 8] about the Newton's first law: "We could hardly sustain that this principle [First law] is a strict experimental result. On the one hand it is not evident how to recognize whether a body is free of forces or not. Even if a unique body in the universe were thought, it is undoubted that its movement could not be rectilinear and uniform in every reference system".

Clearly, to make the law work in embedded engineering, we have to use kludges and patches in our engineering software, and the software will never be robust. Newton's first law violates the

Simultaneity law discussed before, which says everything is interconnected and working simultaneously at same time to make things happen. Newton's all three laws assume isolated environment, which is not feasible in modern embedded engineering. Thus all three Newton's laws are invalid for engineering.

### 4.4 Newton's Second Law

In terms of mathematical notations this simple law is expressed as:

$$f = m * a \tag{8}$$

Here f is the net force acting on an object of mass m. And the resultant acceleration of the object is a. Assuming that the mass is one, and replacing the acceleration by the second derivative of position x, we can rewrite (8) as in (9).

$$\frac{d^2x}{dt^2} = f \tag{9}$$

The first level of modifications that engineers have added to the right hand side (RHS) is the gravitational force g:

$$\frac{d^2x}{dt^2} = f + g(x) \tag{10}$$

In (10) we write g as a function of x, because gravity depends on the position. Note that the quantity x is a three dimensional vector in space, it has North, East, and Up (NEU) coordinates, near earth. The earth has been modeled as an ellipsoid, like the World Geodetic System 1984, (WGS84), and then extensive formulae has been developed by mathematicians, physicist, and engineers to define g(x) as a function of NEU coordinates. These expressions of g are quite complex and can be found in [12, Ch. 7].

Since the earth is rotating around its axis, there is always a force, called Coriolis force that acts on all objects near the earth space. This force has been shown to be dependent on the velocity of the object. Thus the expression (10) should be modified to (11) following [13, p. 76]. Where Ω is related to the earth's angular velocity, which is a constant. A derivation of the formula can be found at [14].

$$\frac{d^2x}{dt^2} = f - \Omega * \frac{dx}{dt} + g(x) \tag{11}$$

Thus we can see that Newton created a very simple equation (8). Such a simple equation violates the complexity law. Since Newton's original equation did not consider x and dx/dt in the RHS of expression (11) we can safely say that the Newton's formula cannot satisfy modern requirements. The second law makes the same assumption as that of the first law. It assumes an isolated environment [15, p. 114], which does not exist anywhere in the universe. It should be pointed out that even after significant modifications of (11) engineers were still not satisfied with the requirements. At present navigation engineers use the help of GPS (global positioning system) to augment inertial navigation system [16] based on (11). We must satisfy all four laws of embedded systems simultaneously with valid design concept without any flaws. At this time we do not have any theory to accomplish that.

It is not necessary to know the modified or better formula to express nature. For, there is no end of improvement in any formula. At the same time our engineering requirements are also becoming more and more sophisticated. We must know that nature is very complex because of GST, and such simple things, like (11), cannot represent nature. If we realize this concept to start with then we will automatically rethink engineering design.

Any assumptions in theory may cause many kinds of problem in embedded system. Most important concern is the stability or convergence of such algorithms. Most of such theories do not

have any proof of convergence in the environment of RTOS and under the boundedness law. As a result the solution that we may get can be completely wrong. Note that the approximation (9) is valid for (11) only under isolated environment that neglects all the new laws presented. Expression (9) says that the object will travel in a straight line, whereas (11) says the object cannot fly in a straight line. Thus application of (9) may cause severe problems even in moderately complex systems.

Similarly observe that (6) has ILT part as its first term. In some sense it can be said that ILT is an approximation of the FLT. But we can see that this kind of approximation can lead to significant distortion of theory, concept, and philosophy. In the case of ILT this approximation moves us from analytic functions to a function of many poles.

### 4.5 Newton's Third Law

All Newton's laws make the assumption of isolated environment or absence of any interactions from other forces. In a sense they are valid only for point particles with nothing else in the neighborhood as pointed out in [17]. We should recognize that nature does not make any assumptions. Our real time embedded systems also cannot make any assumptions because they interact with nature and they have to work.

This third law can be found in [15, p.120] and has been explained in the following way. The forces always occur in pairs or that a single isolated force cannot exist. Any one of these two forces can be called the action force, and the other one then can be called the reaction force. The reaction force is equal in magnitude of the action force and of opposite in direction. Thus the sum of the two forces is always zero and can be written as in (12):

$$F_1 + F_2 = 0 \tag{12}$$

Since we are not isolated, for every action there will always be more than one reaction {$F_2$, $F_3$, … $F_N$}. However the summation of all reactions must still be equal to the original action that produced all the reactions. Therefore in real life we should have (13) instead:

$$F_1 = -(F_2 + F_3 + \cdots + F_N) \tag{13}$$

For example, if you throw a stone, the stone will react with the air particles and will generate a chain of reactions, as it moves in its path. Note that the stone also reacts with the gravitational force and changes its path accordingly. The gravitational force is always there and will react with everything and produce different reactions. The stone will eventually hit a place and create many reactions there. Practically, therefore we cannot have a single reaction. Thus action and reaction always occur in pair as in (12) is not realistic.

Design based on (13) or based on the acknowledgement of simultaneity will result in a new approach to embedded system design. At this time all our mathematical and scientific theories are based on isolated environment. Expression (13) shows that the summation of all forces is equal to zero. Thus the third law is essentially a law of conservation [18]. Thus all we need is to follow the law of conservation.

### 4.6 Kalman Philosophy

The following description of the evolution of Kalman Filter (KF) theory will demonstrate how our mathematical theories are inconsistent with nature and embedded systems. That is, this historical experience for over thirty years has shown that KF cannot work in embedded systems. Alternatively, this history in a sense validates the existence of the laws of nature mentioned in this paper and their use by embedded systems and also indicates the need for a different philosophy.

Many engineering systems can be described by their differential equation models, which can be generated using system theory [19]. These differential equations have several important components: (a) the equation itself, (b) the control variable, (c) the initial value constant and (d)

the parameters. All of them must be known precisely for us to find the precise solution of the system. Kalman said that both control variable and the initial conditions are never known correctly, and he was right. Therefore he proposed a method, which is now known as KF. He proposed to take the measurements from the system and then use them to estimate the solution precisely. There are many books on the subject now, see for example [20].

Very soon engineers realized that the KF does not work. The assumptions behind this theory are not valid for engineering. Most important assumptions that KF violates are: (a) the theory is proven for linear systems only. Since all engineering systems are designed to satisfy the boundedness law, which forces them to become nonlinear, the KF cannot work. (b) Every time slice, the operating system switches the tasks and modifies the environment. The original environment does not exist anymore; when the KF takes over in the next time slice. The implementation of KF does not know and cannot know how the software and the system have changed the GST. (c) Just like control variable and initial value constant are not known for many reasons, the parameters of the equations are also not known for the same reasons.

KF uses the following mathematical equations:

$$\frac{dx}{dt} = A(t)x(t) + B(t)u(t) \tag{14}$$

$$x(0) = x_0 \tag{15}$$

$$y(t) = C(t)x(t) + v(t) \tag{16}$$

In the above equations x(t) describes all the system state variables, like pressure, flow rate etc. The variable u(t) is the control variable, $x_0$ is the initial value constant for x(t) at time zero. The symbol y(t) in (16) represents the measurement and the measurement error is represented by the symbol v(t).

The engineering experiences show that not only the control variable u(t) in (14), and the initial condition $x_0$ in (15) are unknown, the parameters A, B, and C are also unknown, and for the same reasons. There are other KF related matrices like P, Q, and R which are covariance matrices, and are not introduced here for simplicity. But they are also unknown because of similar reasons. In addition the GST effect of the operating system causes more uncertainty for these parameters. Thus the whole KF theory failed to produce results correctly and required lot of kludges in the implementation of software.

The engineers therefore decided to introduce adaptive KF (AKF) that will continuously estimate ABC and PQR matrices. There are many papers that have been published over long period of time on AKF theories. They have given detailed theoretical proofs on the convergence and optimality of the AKF. Some results can be found here [21], and [22]. The very fact that we have started using methods to estimate parameters is a recognition that the KF is not working, and the assumptions of KF are wrong. The AKF did not work also because of the boundedness law and the GST effect. The AKF uses a linear model.

The engineers then extended the theory to nonlinear equations, known as the extended Kalman filter (EKF). So far no one has produced any theoretical proof that EKF is valid. That is, no one has proved that the EKF filtering equations will converge; will provide optimal solution etc. If we use something that does not even have a theoretical foundation under its assumptions, then that theory cannot work at all in embedded engineering. The paper [23] writes that the EKF fails to account for fully nonlinear dynamic system. The paper [24] says for nonlinear noisy systems optimality has not been proven. But these are not enough; the KF must satisfy all four laws of embedded systems to qualify for its usability; and it does not.

### 4.7 Differential Equations

Now we show that the theory of differential equations (DE) cannot work in our technology. The linear differential equations like (14-16) cannot work in any embedded engineering systems, because all embedded systems are nonlinear; since by design systems implement the boundedness law. All digital systems are discrete in both time and in space. Therefore, there cannot exist the concept of limit, continuity, and derivatives, and so no nonlinear DE will be meaningful for embedded engineering.

When we approximate derivatives by making it discrete in time and space then we have to create complete theory for their existence, uniqueness, convergence, optimality etc. for all nonlinear discrete systems that include the boundedness law. There exist some theories, only in a limited sense, for continuous time systems [25], but not for discrete cases. We have expressed these concerns while discussing EKF case. If there is no theory then that idea should not be used in any applications. It will create more confusion, more problems, more patches, kludges, and provide poor performances.

Today our technology is processing communication systems even at 10 or 20 GHz rate using digital technology [26] and using parallel processing methods. As the technology evolves we may see even faster rate sampling of analog signals and use of digital techniques. Although it is possible to implement the digital ideas in [26] using the theory described in [4] and implementing by analog means as presented in [27], it is still not conceivable that there will be a need for DE in embedded systems in near future.

We should also realize that the derivative is actually a law of conservation in disguise [18]. The derivative says that a small change in one thing will produce a small change in another thing. According to the action-reaction law, these two changes must be same also. Therefore Δy, a small reaction in something, say y, divided by Δx, a similarly small action on something else, say x, must always be one. We can then write as in (17):

$$\frac{\Delta y}{\Delta x} = 1 \tag{17}$$

We are neglecting the sign for simplification, and without loss of generality. This ratio is called the derivative. When these delta values are very small, the ratio is denoted as dy/dx, and can be represented as in (18).

$$\frac{dy}{dx} = 1 \tag{18}$$

But in the theory of differential equations the derivatives are not always equal to one. There are many reasons for that. This happens mainly because x and y are normally not the same kind of variables or do not have the same units. If we convert both variables to the same physical unit, then the derivative will always be equal to one, because this was derived from the action reaction law (8), the third law of Newton. Thus in all of our modeling approach we should always ensure that (18) holds. Therefore by introducing differential equation we cannot get any new characteristics in our models. Best thing will be to follow the law of conservation moment by moment.

A theory to control continuous time dynamic systems using DE and with boundedness nonlinearity can be found in paper [28] and in [3]. The idea in the paper definitely solves an important class of embedded engineering problems. The paper replaces the nonlinear function of the boundedness law by a smooth tanh trigonometric function and then uses Hamilton Jacobi Bellman method to solve the control problem. However, a realistic implementation of the idea for a complex embedded project operating under multitasking operating system is yet to be

demonstrated. This author does not feel confident that it can be implemented for discrete systems using a kludge free software.

### 4.8 Measurable Functions

Many theories in math, science, and engineering use the assumption of piece wise continuous (PWC) functions. These functions have discrete jumps in certain places over their domain. There are no piece wise continuous functions in any embedded engineering application. If you put an oscilloscope probe on any pin of a digital microprocessor you will find a continuous signal. Thus even in digital electrical engineering there are no PWC functions.

Measurable functions are another step extension of PWC in the abstract direction. Most common example found in textbooks is the following:

$$f(x) = \begin{cases} 1 & x \text{ is rational} \\ 0 & x \text{ is irrational} \end{cases} \quad \forall\, x \in [0,1] \tag{19}$$

If PWC is nonexistent, then how can (19) be useful in engineering? This kind of functions has been used as foundations of measure theory to extend the theory of integration, theory of differential equations, stochastic process [29] etc. The concept of measure theory provides a foundation of probability theory also.

We have seen that the KF, which is based on probability theory, does not work. No matter how sophisticated theory we make using measureable functions, at the core of it we must find the probability of every event of the process. These events are defined using the set theoretic concept of sigma algebra [30]. It is not possible to find probability of events in a sigma-algebra for a real time engineering systems. No product will be reliable if it is based on such chance events, with assumptions. As mentioned before, nature and therefore engineering do not make any assumptions. Moreover probability is a linear measure also and engineering is not. We must extend mathematics downwards for its feasibility in engineering, and not upwards in abstract imaginary fields.

## 5: PROPOSED DESIGN

We have taken several standard and commonly used theories to show that none of them obey the new laws of nature identified in this paper. If we pick any theory from our existing database of theories we will be able to show that none of them will satisfy the new laws. Thus we will make our software unreliable; engineering will remain vulnerable, when we use such theories.

As mentioned before the goal of this paper was to point out the root cause of the reliability problem of embedded systems, however we want to venture little bit into the idea of an approach for the alternative. The suggested approach in this section is quite dramatic, and yet very natural and taken from human activities. It is very difficult to think radical in our community and abandon all existing mathematical methods and go for a new approach. However, we believe that this approach will become a common practice in days ahead, as the sensor technology becomes cheaper, smaller, and better.

If we know that the existing theories cannot work, because they violate all the laws, then why use any kind of theory to begin with. Best approach in engineering will be to use as much measurements as possible, and sample as fast as possible [10], and use them to make decisions at the next sample time. It will be always possible to implement such a design in all engineering problems; we just have to rethink it using a global view of our requirements and technology. Elimination of all theories that violate the laws, will significantly improve the quality of our embedded software. A 10 GHz high speed digital communication system has been proposed [27] to implement such a scheme without using any conventional adaptive algorithms.

The lessons from Kalman's idea are very profound, and that is why we call it a philosophy in this paper. Kalman actually wanted to present his philosophy and provide its theoretical foundations with the help of a linear model. It says use the measurements to estimate what you want. We humans do not use math and science when we walk; we take measurements constantly using all our sensors, like eyes, ears etc., to make our ways around correctly. We have the technology and do not need to use inconsistent theories of math and science in our engineering products. After all our KF experiences show that we cannot find ABC and PQR parameters. This measurement only approach may even reduce the cost of development and maintenance of the product. Engineering will become simpler also, and will be very robust.

Our technology may not be able to provide high quality sensors today at low prices. Under these circumstances we should take advantage of the simultaneity and complexity laws by using multiple sensors at multiple locations and processing them simultaneously using simple algebra. It is also very important to rethink the requirements and isolate it carefully from underlying design assumptions, which often include mathematics and science as its foundation. In the design process, while working at the details level, we must repeatedly come back to our high level thought processes to look at the requirements, in order to avoid such math and science theories.

In summary, we use theories only because we do not have enough measurements, the measuring devices are not small, and are expensive. To compensate for the lack of these measurement data, we use unproven theories and patch work to extrapolate and interpolate data. If we rethink engineering then we will invest and produce better, smaller, and low cost sensors to make large set of simultaneous and interactive measurements thus eliminating the theories completely.

## 6. CONCLUSIONS

We have shown inconsistencies in many of our mathematical and scientific theories with the natural laws of embedded systems. If we take any theory from math and science, and then look at it using the embedded laws, we will be able to show its invalidity. Because of these problems, these theories cannot be used in embedded engineering. If we use them then we will be forced to add many patches and kludges in our embedded real time software to make engineering work and that will make our software very unpredictable, difficult to maintain, test and debug. It is possible to rethink the entire engineering architecture and design problem using the Kalman philosophy of taking measurements, and using very simple algebraic theories.

## 7. REFERENCES

[1] M. Guido. and B. Andrich. "Metric suite for directing the failue mode analysis of embedded software systems", ICEIS, 2005

[2] H. Kopetz. "The complexity challenge in embedded system design", Research report 55, Austria, 2007

[3] S. E. Lyshevski. *Control system theory with engineering applications.* Boston, USA: Birkhauser, 2001, Chapter 4.

[4] S. Das, N. Mohanty, and A. Singh. "Function modulation – the theory for green modem", pp. 121-143, International journal on advances in networks and services, vol. 2, no. 2&3, 2009

[5] R. Boynton."Precise measurement of mass", Soc. allied weight engineers, Arlington, Tx, 2001.

[6] Discovery,(2007), Neurons, How they work, Discovery channel video, http://www.youtube.com/watch?v=c5cab4hgmoE

[7] C. Halls. et al. “The need to reintegrate the natural science into economics”, available: http://dieoff.org/page228.pdf , 2001

[8] L. Debnath. *Integral transforms and their applications*, CRC press, 1995

[9] S. Das. “Finite time engineering”, WCECS, San Francisco, October 19-21, 2011

[10] S. Das. Mohanty, N. & Singh, A.: “Is the Nyquist rate enough?”, ICDT, IEEE Computer society, available: IEEE Xplore digital library, 2008

[11] R. Ferraro. *Einstein’s Space-time*, Springer, 2007

[12] A. B. Chatfield. *Fundamentals of high accuracy inertial navigation.* Virginia, USA: Progress in Astronautics and Aeronautics, 1997, Volume 174.

[13] R. M. Roger. *Applied mathematics in integrated navigation systems*, second edition, Virginia, USA: AIAA, 2003, p. 76.

[14] R. Christensen. and N. Fogh. *Inertial navigation systems*, masters project, department of electronic systems, Aalborg University, Denmark, available: http://www.control.aau.dk/uav/reports/08gr1030a/08gr1030a_student_report.pdf , 2008.

[15] R. A. Serway. and J. W. Jewett. *Physics for scientists and engineers with modern physics*, Sixth edition, USA: Thomson, 2004.

[16] M. Nylund. A. Brown. and J. J. Clark. “Kinematic GPS-Inertial navigation on a tactical fighter”, Proc. ION GPS, Oregon, USA, 2003

[17] M. J. Pinheiro. “On Newton’s third law and its symmetry breaking effects”, Phys. Scr. 84 055004, 2011

[18] S. Das. “Laws of conservation”, Intellectbase academic conference, Las Vegas, Dec 15-17, 2011

[19] D. Rowel. “Linear graph modeling: state equation formulation”, MIT, Department of mechanical engineering, advanced system dynamics and control, available: http://web.mit.edu/2.151/www/Handouts/EqFormulation.pdf , 2004

[20] M.S. Grewal. and P.A. Andrews. *Kalman filtering: theory and practice*, NY: John Wiley, 2001.

[21] M. Oussalah. and D. J. Schutter. “Adaptive kalman filter for noise identification”, 25th international conference in noise and vibration engineering, Belgium, 2000.

[22] K. Xiong. et al. “Adaptive robust extended kalman filter”, From the book – Recent advances and applications, I-Tech, Vienna, Austria, 2009

[23] S. Gillijns. et al. “What is the ensemble Kalman filter and how does it work?”, Proc. Am. Cont. Conf., June, MN, USA, 2006

[24] M. Boutayeb. et al. “Convergence analysis of the extended kalman filter as an observer for non-linear discrete-time systems”, Proc. 34th. Conf. Dec. & Con., New Orleans, LA, December 1995

[25] H. K. Khalil. *Nonlinear systems*, third edition, New Jersey, USA: Prentice hall, 2002.

[26] Broadcom, US Patent, “Electronic dispersion compensation utilizing interleaved architecture and channel identification for assisting timing recovery”. US Patent number – 7830987, Nov 9, 2010

[27] S. Das. “A new analog design for the EDC family”, Vitesse, California, USA, internal proprietary and private document, 2011

[28] S. Lyshevski. “Constrained optimization and control of nonlinear systems: new results in optimal control”, 35 th conference on decision and control, Kobe, Japan, 1996

[29] W. Wong. And B. Hajek. *Stochastic process in engineering systems.* NY, USA: Springer-Verlag, 1985.

[30] P. R. Halmos. *Measure Theory*. NY, USA: Springer-Verlag, 1974.