The FUN Theorem

(The FUNdamental Equation of Space-Time Project)

by Woody Stanford, (c) Stanford Systems 12/18/2016

Note: this is a preliminary draft covering only the theorem. The implementation of the theorem as a computer algorithm will be expanded upon in the actual version.

Introduction to the Theorem

Axion-based research is all the rage these days; basically particles of space. The FUN approach is interesting in that it approaches the axion problem from the point of view of a classical Einstein interpretation.

We think that the difficulty with current directions of axion-based research is that it doesn't use the emerging entanglement paradigm. To being into the mix classical Einstein curved space we think we might have a winner with the springs involved being introduced in the perpendicular direction (rather than like springs in a mattress.

How the Theorem Works

How the theorem works is simple. Let us explain...

The theorem in one-dimension explains it best I think. Its based on some axioms, such as:

(1) The Key of Space: that the key to understanding reality is that if you understand the behavior and fundamental nature of a single point in space that you understand all other points in space (its that homogeneous).

(2) A Counter-Intuitive Axiom: that a point in space is ONLY influenced by the points IMMEDIATELY around it.

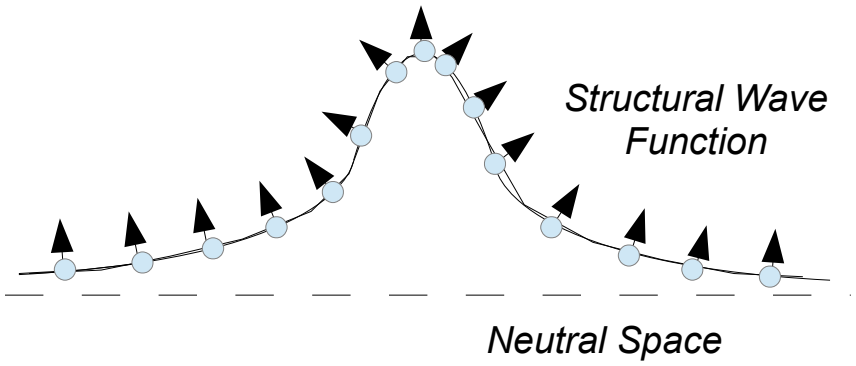
(3) Spacial Curvature: that space is fundamentally curved (the second derivative of its slope at any given point) even, and especially at subatomic scales, as it is at macroscopic scales. That this curvature results in a mass, space's fundamental resistance to mass/energy.

The one-dimensional time-invariant version can be worked out by a computer algorithmically. The time invariant-version merely expresses the point forces as forces only; its only through translating, animating those forces in real-time that the theorem becomes time-variant.

The One-Dimensional Time-Invariant and Variant Versions

The time-invariant version takes the forces involved and just leaves them as vector quantities, where as the time-variant version animates the spacial sheet according to the resistance of the sheet and the force vectors involved.

Pictorially, this is what we are doing algorithmically:

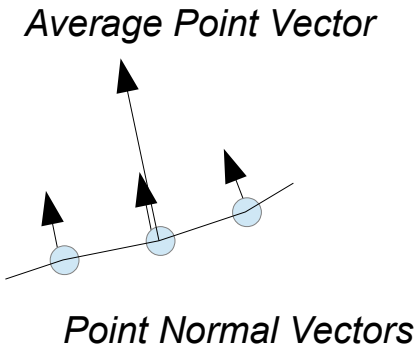


Each arrow represents a 2-dimensional vector: a spacial component, with a direction component so mathematically:

$$\vec{pnv}(x,\theta)$$

PNV stands for *point normal vector*, which is a unit vector, so its amplitude is always unity. The reason for this is that every point in space is identical to all others. What we do is we constantly get the sheet to try to "straighten out". This requires an algorithm that implements the FUN algorithm. In one-dimension the approach is straight forward. What is done is we take the adjacent points on either side of a point and calculate the average point vector.

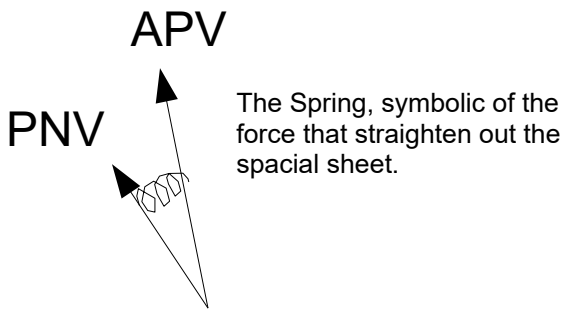
Pictorially this is what we do:



We basically sum the three vectors and divide by 3 to get the average point vector:

$$\vec{apv} = \frac{\sum \vec{puv}}{3} \quad .$$

Then we attach the spring:



The spring is symbolic of the force that tries to “correct” the point normal vector against the average point vector. This force is a fundamental constant based on space's fundamental resistance to curvature, we are predicting proportional to C.

This is where the time-invariant version stops (it just expresses the relationship between the PNV and the APV in terms of a force).

The time-variant version is performed via an iterating computer program that animates the spacial line or sheet or volume, translating the force into a point correction translated through the PNV's like they were dowels. This has the effect of straightening out the sheet. The real trick is in developing structures on the sheet that can “stand” or in other words support matter waves....particles.

The Two and Three-Dimensional Versions

The two-dimensional version is the same algorithm just upgraded to 2 dimensions. This is the version (with time-variance) that we will use in the FUN simulator (we have it 75% coded at the time of this writing). We hope to initially show photons as propagating regions of the sheet that radiate mild waves in a given direction. We hope to introduce EXPER-type structural wave functions and see if they actually “stand” in this kind of simulator.

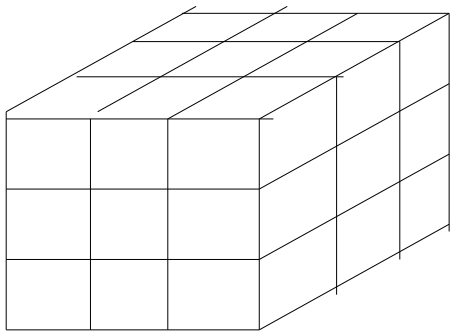
We also hope, with various kinds of “initial conditions” (basically rough wave forms that we drop onto the sheet to start the simulation at) to see little meson-like formation of varying duration moving away from the initial wave forms. If our simulation can show these two phenomenon, we will consider it a success and an important step towards Grand Unification.

The two-dimensional version is performed by way of what we call a “tic-toe”. There are 8 adjacent points around a given spacial point in 2D, like this:

p1	p2	p3
p4	p	p5
p6	p7	p8

…with p being the pertinent point. Points 1 through 8 are the adjacent points. We are hypothesizing that the nature of a single point in space can be fully modeled by its interaction with the points directly around it. It is through the propagation of a matter or energy wave that it affects other points, not directly at a distance. Counter-intuitive to be sure, however we think this is how space actually works.

For completeness's sake let's cover how the 3D version works, because with the success of the FUN simulator, it is merely a question of upgrading it to 3D and then running it with structural wave functions hypothesis. It is done with a 3D “tac-toe” like this:



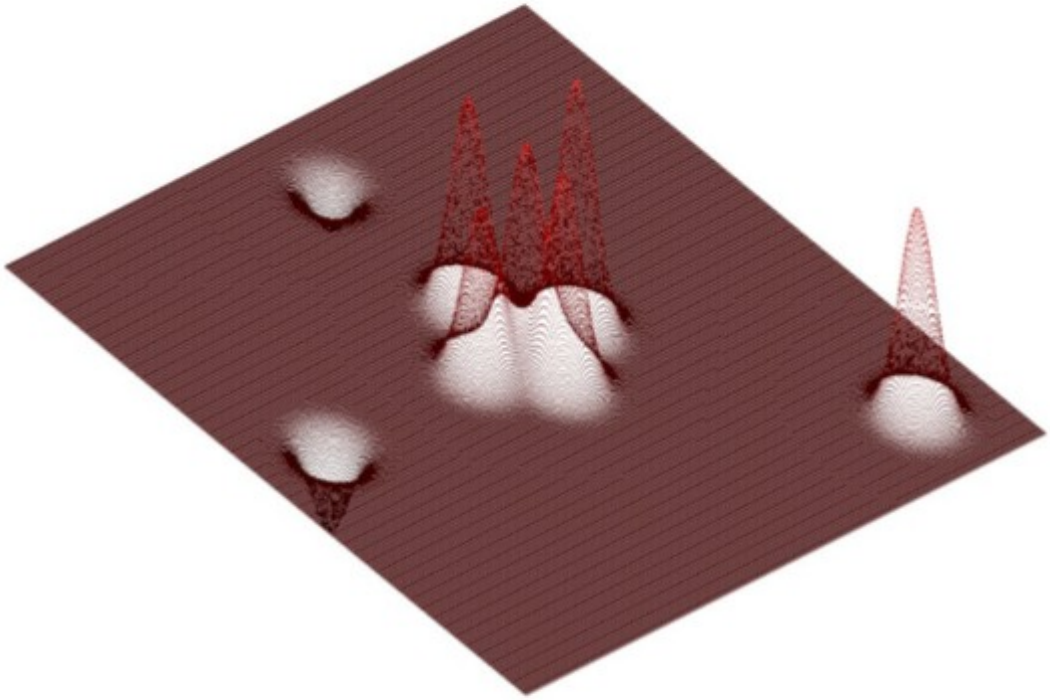
The center of each subcube in this tac-toe is a spacial point with the center of all of them being the pertinent point. The 26 points around it (3 x 9 – 1) are what influence it's “behavior” according to the algorythm described earlier.

The EXCITING PART is that this should fully simulate space and time when done in time variance when done in 3D (with an amplitude vector that describes the spacial 'brane). So with a total of five-dimensions we are hoping to describe successfully what takes string theory 11-dimensions to model. We basically take the complexity of 11 dimensional vector space and map it to a 5-dimensional paradigm with the vector complexity (it's quantum information) translated to geometric/topological complexity.

Seriously, we think reality is this simple. The difficulty is in modeling the correct wave functions that cause the simulation to work. We could just drop in our models from EXPER and find the right constant of spacial resistance and it should just model reality. Instead of large, expensive colliders we can just simulated on workstation or supercomputer, a holy grail of modern physics.

Diagrams from the FUN Simulator

The FUN simulator isn't quite finished at the time of writing so we can't show you actual outputs of the simulator, but we can show you its graphics support, and representative initial condition states, and what we expect to see out of it.



This is what we call the “rubber sheet” that implements the FUN theorem as a computer simulation. The wave forms on the sheet are put there by the initial condition generator. What we expect to see is that they immediately collapse. What we want to see is that during the collapse that they cause radiating ripples in the sheet, which would be indicative of photons, light.

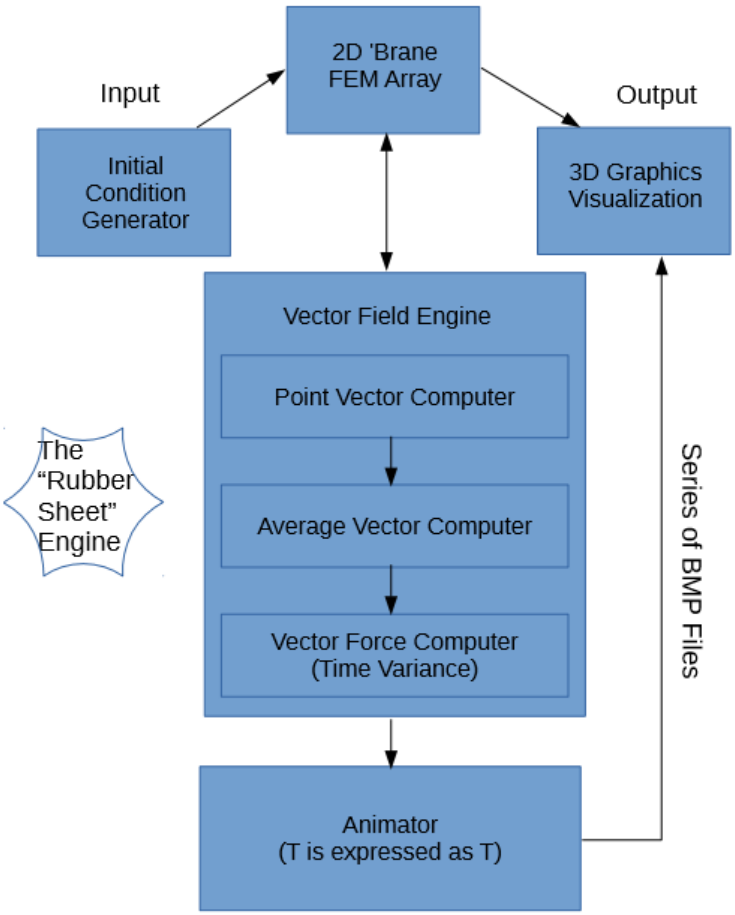
If we are really lucky we will see moving “particles” (albeit 2D ones) that indicate the sheet's ability to sustain standing waves, or matter. We doubt if they will be identical to the 3D versions, however we hope to capture a few shots of them and reconcile them with our EXPER models and see if there are any similarities.

An Issue: Boundary Conditions

An issue right now is how to code the boundary conditions such that it acts like a sheet much bigger. Once a moving wave form moves off the sheet, the program no longer tracks it, so this concern will have to be addressed.

Coding Examples with Explanations

Here are some of the operative code snippets that explain the main points of its operation. However the complete simulator can be expressed in the following block diagram.



The main program component is the 2D Brane FEM Array, that expresses an area of microspace in terms of a Gaussian surface which is modeled with scalar double-precision floats. The Vector Field Engine is what actually computes a field of PNV's and its associated AVC's which are then calculated to the by the Vector Force Computer to yield the force values that are used to modify the FEM array (in the "Animator" component) simulating time passing.

This is actual time passing, not like EXPER where we merely iterated the position of simulated particle models.

We can pass the results out by data file, but its so easily graphically interpreted that we have our 3D graphics support that outputs the result of the simulation as a short animation (expressed as a series of BMP file frames). We then view the results of the simulation in a slideshow utility which allos is to with our mouse roller to pan forward and back stopping right at points of interest.