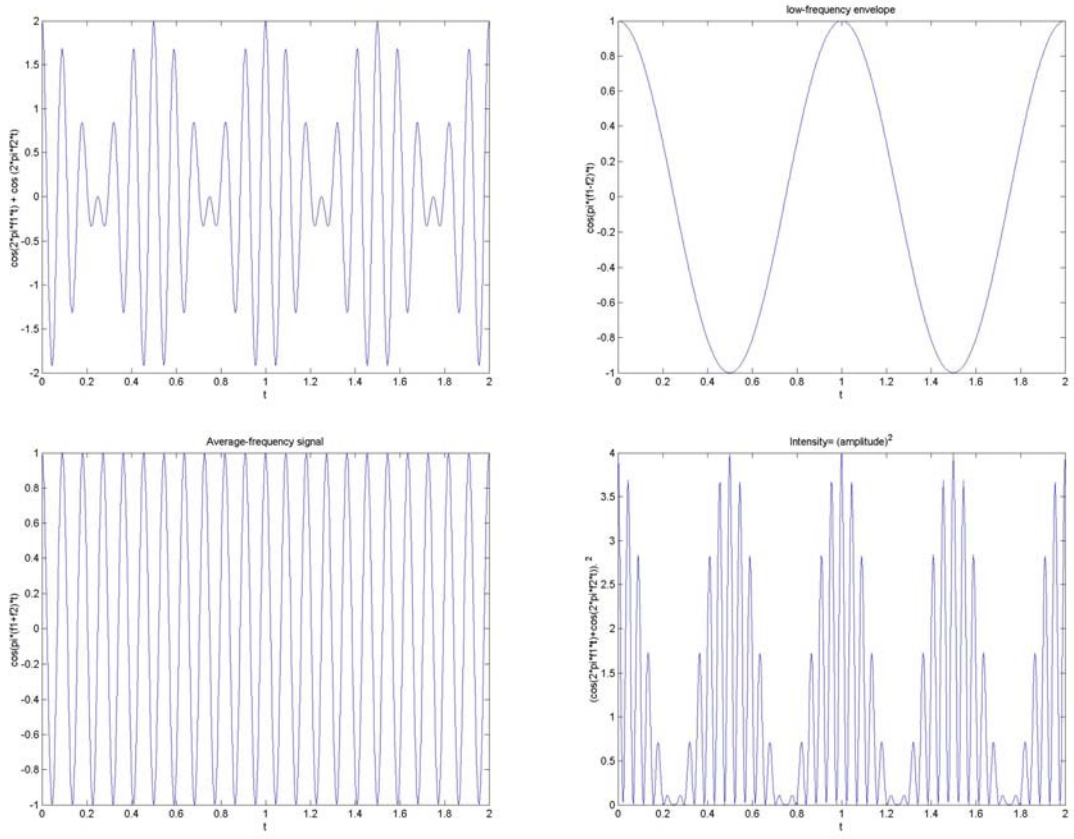
Meeting #8

What is design? A design example. In the last meeting we saw an example of a design problem (vehicles for the peak performance competition) with multiple solutions from the different teams, which can be grouped in two categories: offensive strategies using blade power and catepillar wheels and defensive stragegies using wedged shape, to have the opponent ride off vertically or horizontally. When additional specifications are given such as one 9V battery, maximum 2 kg vehicle, etc (see A design example link or read the DCE text) then one type of design will carry an advantage over the other. When economy is needed, passive-dynamics (such as that employed in a recent model of a walking robot that requires ten times less energy than the Asimo: http://www.news.cornell.edu/releases/Feb05/AAAS.Ruina.bipedal.ws.html) or defensive strategies are preferred over well-equipped and power-consuming offensive strategies. This process of ideas generation, mental testing and scrutiny keeping in mind the specifications, to arrive at the best option that will satisfy the project requirements is a design process. Going back to the previous stage is part of the process, and so the concept of a design cycle we talked about in the previous meeting. Also many times even after a prototype was built and its analysis and testing done, the design needs to be improved by going back to the specifications/mental stage. Check the design cycle in the notes for the previous meeting.

To continue with our design example, answering "**why not making a shorter vehicle"** is a typical task of a design team. This is CW2 which was done today.

As related to Project 2:

What is the beat phenomenon? It occurs when two periodically oscillating signals or sinusoids arrive at a same location and undergoing superposition. According to trigonometry, this sum is equal to a product of a sinusoid whose angular frequency ω (ω=2πf, f is the linear frequency) is the difference of the original frequencies divided by two (this signal is called the **amplitude envelope**), and a second sinusoid whose angular frequency is the average of the original frequencies (I call it the **average-frequency signal**). The **intensity** is the square of the amplitude, which has as frequency twice that of the amplitude envelope, or just ($\omega_1 - \omega_2$). This low frequency oscillation of the intensity is what we hear as the **beats**.



A demonstration was made using a LabVIEW Virtual Instrument.

Hw 1 was assigned. There are three design problems and two LabVIEW questions:

LabVIEW:

-LabVIEW: background for HW1 questions 4 and 5. **How to locate different functions within the Front Panel** (user interface: inputs and outputs quantities) **and within the Block Diagram** (programmer interface: operations, analysis).

-Things belonging to the Front Panel will be found under **Controls palette**, abbreviation is C, which can be brought up by 'right-clicking' (click on the mouse's right button) on the Front Panel.

-Things belonging to the Block Diagram will be found under **Functions palette**, abbreviation is F, which can be brought up by 'right-clicking' on the Block Diagram.

-Things to operate VI (Virtual Instruments) will be found under **Tools palette**, abbreviation is T, which can be brought up by clicking on Window, then select 'Show Tools Palette'

For example, where to locate 'Array'?. We start by guessing whether this is an input/output or an operation, it is more of a numeric input utility, so right-click on the Front Panel to bring up the Controls palette (C), then select subpalette All Controls, then select subpalette Array and Cluster, to find Array in the first button. So the complete location for Array reads C/All Controls/Array and Cluster.

For example where to locate 'Reciprocal'?. This is more of an operation (getting the reciprocal of x is doing 1/x). So right-click on the Block Diagram to bring up the Functions palette (F), then select All Functions, then Numeric, and find Reciprocal under button 1/x. So the location for Reciprocal is F/All Functions/Numeric

-As a background to changing values on a Numeric Control, we make a simple addition VI. We will need two inputs (Numeric Controls) and one output (Numeric Indicator) in the Front Panel, where the user will input the numbers she/he would like to add, where she/he will read the result, respectively. We should label the inputs as n1 and n2, and the output as n1+n2. This is necessary to identify identical elements on the Block Diagram, and as part of the user interface. **Label** can be entered by typing into the blank box that is shown when an element is placed in the Front Panel. The blank box can be brought up by right-click on the element and select '**Show Label'**. To edit labels, select the Text tool, under Tools Palette, then click on the label.

To tell LabVIEW how to produce the output from these inputs, the programmer goes to the Block Diagram to place the addition operation (F/All Functions/Numeric), then wire the inputs n1 and n2 into the left terminals of the addition by selecting the **wiring too**l (under T), click to start, click to end, double-click to finish a wiring. Note that since the addition is commutative, n1 and n2 can be individually connected to either the left upper terminal or the left lower terminal of the addition operator. Should we have a division, the upper terminal is divided by the lower terminal, or the left terminals are different. In a subtraction, the upper terminal is subtracted by the lower one.

If we need to remove any piece of wire, use the **Select tool** (arrow under T) to select that piece, then hit 'backspace' on the keyboard.

Then it comes to put in values into the Numeric Controls to test our addition VI: using the **Operate Value tool** (finger, under T) and click on the left handles to increase or decrease by an integer unit. To run the VI, click on the **Run button** (right arrow in the upper left corner), the results should show as expected. If we would like to scan through different inputs and outputs without having to hit the Run button every time we change the inputs, then use **Run Continuously** (found to the right of the Run button). Under this mode, the VI should be stopped before any modification can be made.