Why should I use VisSim to enhance my Mathcad engineer skills?

Référence :E:\0FB\MBI\Projets commercialisés\VisSim\Action commerciale\Pub\Pub VisSim Boston\Comparaison Mathcad VisSim

About the author:
Dr Pr François Bernot is professor at the university of Tours (France) with expertise in the field of power electronics and motor control & design. He supports **VisSim** in France, giving trainings for industrial customers. He also gives **Mathcad** trainings and applied engineering trainings in his expertise field. He wrote a book on power electronics motor drives and several scientific papers. He speaks fluent french, english and spanish and **can give trainings** all other the world.
Contact: francois.bernot@wanadoo.fr (33) 6 75 90 32 82

Introduction

I am a regular user of **Mathcad** for my industrial scientific expert activity. I discovered **VisSim** several years ago, and I was surprised by its complementarity to **Mathcad**. Indeed, this last software is very interesting, because it allows the engineer in a same computation sheet, to write the specifications of the given problem, to solve it and therefore to simulate it.

Unfortunatly, **Mathcad** is not designed for real-time interactive simulation. It is possible to overpass this limitation, I tried it. But I confess that **VisSim** give a confortable response within a reduced time, while **Mathcad** needs more time of development.

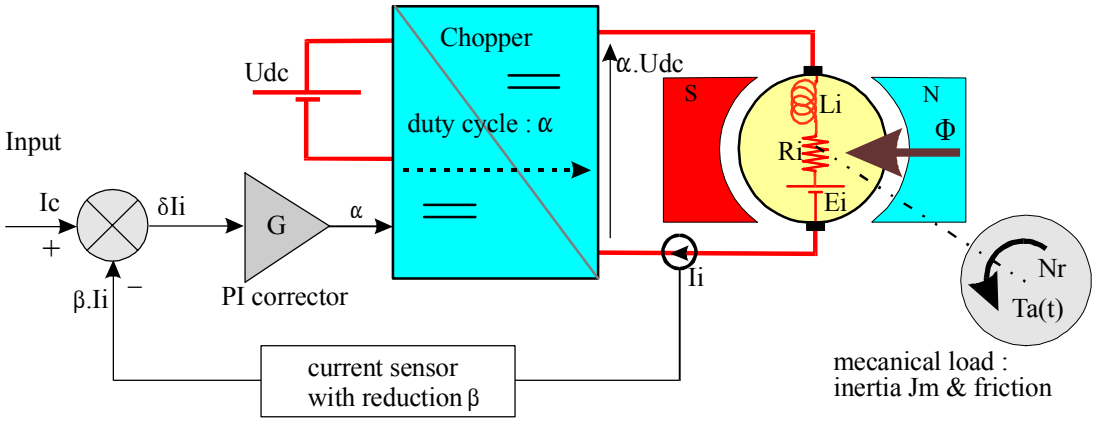
I can assess that **Mathcad** is a very good support to prepare **VisSim** work. It gives a good support for the description of the equations, combining images, text and equations. **VisSim** allows to go further, providing an interactive real time simulation with true non linear ability. What costed me a lot of time with **Mathcad** alone, like non-linearities and interactive simulation, became an easy step with **VisSim.**

Let us now deal with the analysis of the discussed device: a chopper controling a PM brush motor. I meet it very often in my engineer Job. I give it also to my engineer students at my university. It is quite an interesting problem, not so obvious to analyse.

Combining **VisSim** and **Mathcad**, its complete study costs only one full day for me, or 4 hours of exercises and 4 hours of practicals for my students. We also use dedicated power bench powered by **VisSim** that allow to download the simulation onto a DSP controlled power inverter.

$$Ta(t) = Jm.d\Omega r/dt + Tr + Tv.\Omega r./\Omega rn = k\Phi.Ii$$

$$Ei = k\Phi.\Omega r$$



All the datas and functions used in this sheet are given at the end of this document. It can be runned with all Mahtcad versions older than 2001i. Take care to place the function **Mathcad** documents *Control of a PM brush motor functions.mcd* in the same folder as this one.

Simulation of the motor alone

Modelizing the PM brush motor leads to the following set of equations:

$$Udc(t) = Ei(t) + Ri \cdot Ii(t) + Li \cdot \frac{d}{dt} Ii(t)$$

$$Ta(t) = Jm \cdot \frac{d}{dt} 2\pi Nr(t) + Cro(t)$$

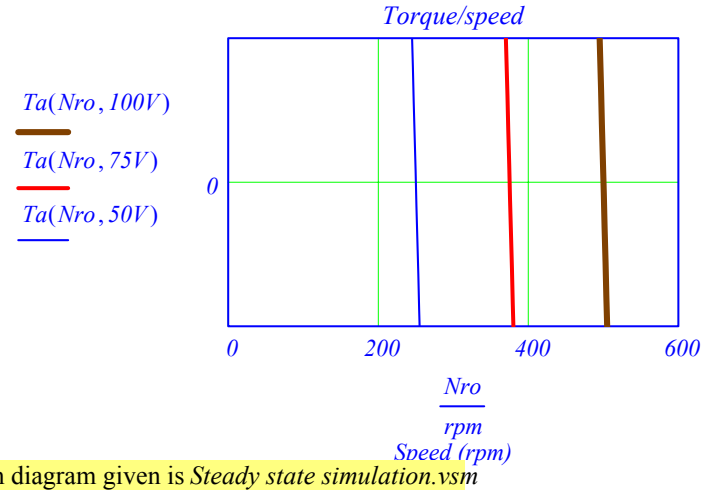
Steady state simulation

Studying the motor in seaddy state behaviour is quite easy and **Mathcad** is a very good support, even if non-linearities appear.

Let us define the rotor transformation coefficient: $k\phi := \frac{Ein}{(2\pi Nrn)}$

the nominal torque $Tan := k\phi \cdot Iin$

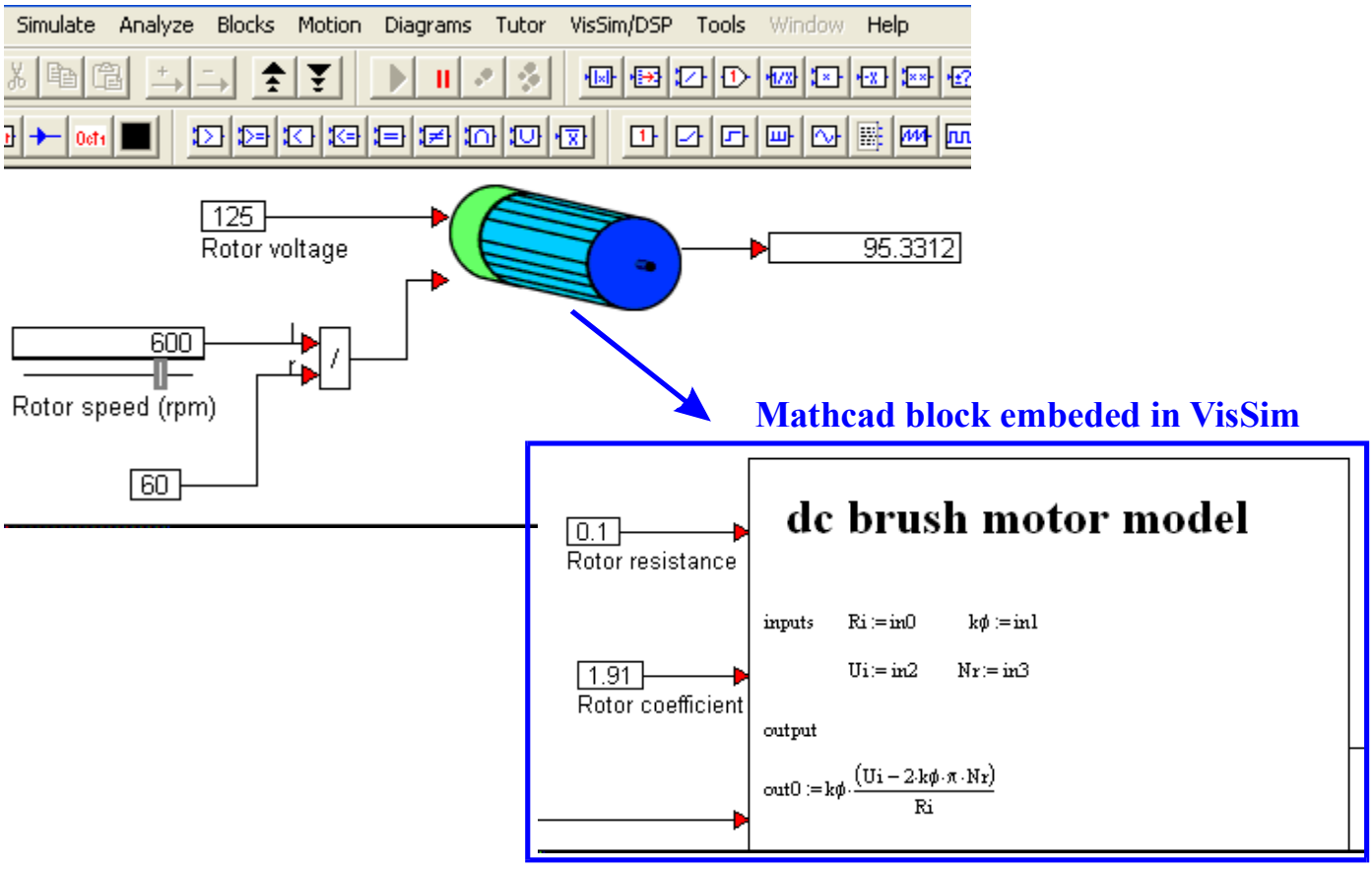
and the torque function $Ta(Nr, Ui) := k\phi \cdot \frac{(Ui - 2 \cdot k\phi \cdot \pi \cdot Nr)}{Ri}$



The user at this step can confront these results to its **VisSim** dynamic model. Doing so, he can transfer directly his **Mathcad** model to a **VisSim** box.

Take for example $Ta(600 \cdot rpm, 125\,V) = 95.493\ N \cdot m$ in the next **VisSim** diagram.

VisSim simulation including Mathcad embeded block: VisSim diagram given is *Steady state simulation.vsm*



Transient simulation with Mathcad

Let us now look for the transient behaviour, I have poor **Mathcad** options: Laplace transform or incremental simulation.

Laplace transformation modelisation uses the following process. It is very interesting, because it give analytical formulas (of wich real use?). **Mathcad** is a very good tool for this work. It helps me a lot, because I cannot remember all the transformation formulas.

The electrical equation $Udc(t) = Ei(t) + Ri \cdot Ii(t) + Li \cdot \frac{d}{dt} Ii(t)$ gives in the Laplace space $\frac{(Udc(s) - Ei(s) + Ii\_init \cdot Li)}{(Ri + s \cdot Li)} = Ii(s)$

The mecanical equation $Ta(t) = Jm \cdot \frac{d}{dt} 2\pi Nr(t) + Cro$ gives in Laplace space

$$Ta(s) = 2 \cdot Jm \cdot \pi \cdot (s \cdot Nr(s) - Nr\_init) + \frac{Cro}{s} = k\phi \cdot Ii(s)$$

Mathcad gives you full support, to compute the following speed equation, for the step response:

$$\frac{1}{2} \cdot \frac{\left(2 \cdot Jm \cdot \pi \cdot s \cdot Nr\_init \cdot Ri + 2 \cdot Jm \cdot \pi \cdot s^2 \cdot Nr\_init \cdot Li - Cro \cdot Ri - Cro \cdot s \cdot Li + k\phi \cdot Udco + k\phi \cdot Ii\_init \cdot Li \cdot s\right)}{\pi \cdot s \cdot \left(Jm \cdot s \cdot Ri + Jm \cdot s^2 \cdot Li + k\phi^2\right)} = Nr(s)$$

If your drugstore is opened you can try to eat vitamins, in order to include non-linear behaviour in the previous equation, I could not. But for the moment, let us find the inverse Laplace transform solution. In this case, I was lucky, because **Mathcad** gave me an interesting solution. It is quite longer than what I expected, but it exists. It is given in the associated function **Mathcad** file (*Control of a PM brush motor functions.mcd*) with the name $Nr\_Laplace\_f$.

Incremental simulation

While you do not have installed **VisSim** on your computer (full free demo at http://www.vissim.com/downloads/demos.html), you can perform true non-linear simulation. It could be real time, but with a little delay, because every change costs a lot of vitamins. A a consequence, if your drugstore is closed, it is easier to download the instal **VisSim** file and therefore to order it.

Considering dt as Δt, and d as Δ, it is possible to transform the previous electrical and mechanical equations as (n is the index describing the time):

$$\begin{cases} Udc_n = Ei_n + Ri \cdot Ii_n + Li \cdot \frac{\Delta Ii_n}{\Delta t} \\ Ta_n = 2\pi \cdot Jm \cdot \frac{\Delta Nr_n}{\Delta t} + Cro_n \end{cases} \text{ or better } \begin{cases} Udc_n = Ei_n + Ri \cdot Ii_n + Li \cdot \frac{Ii_n - Ii_{n-1}}{\Delta t} \\ Ta_n = 2\pi \cdot Jm \cdot \frac{Nr_{n+1} - Nr_n}{\Delta t} + Cro_n \end{cases}$$

Combining those two equations leads to the intermediate solution:

$$Udc_n = k\phi \cdot 2 \cdot \pi \cdot Nr_n + \frac{1}{k\phi} \cdot \left[\left(Ri + \frac{Li}{\Delta t} \right) \cdot \left(2\pi \cdot Jm \cdot \frac{Nr_{n+1} - Nr_n}{\Delta t} + Cro_n \right) - \frac{Li}{\Delta t} \cdot \left(2\pi \cdot Jm \cdot \frac{Nr_n - Nr_{n-1}}{\Delta t} + Cro_{n-1} \right) \right]$$

Finaly, the speed Nr_{n+1} at instant $n + 1$ is obtained:

$$Nr_{n+1} = \frac{1}{2} \cdot \frac{\left[\begin{array}{l} \left(Udc_n \cdot k\phi \cdot \Delta t^2 - 2 \cdot k\phi^2 \cdot \pi \cdot Nr_n \cdot \Delta t^2 \right) \ldots \\ + \left(2 \cdot Ri \cdot \Delta t \cdot \pi \cdot Jm \cdot Nr_n - Ri \cdot \Delta t^2 \cdot Cro_n \right) \ldots \\ + 4 \cdot Li \cdot \pi \cdot Jm \cdot Nr_n - Li \cdot Cro_n \cdot \Delta t - 2 \cdot Li \cdot \pi \cdot Jm \cdot Nr_{n-1} + Li \cdot Cro_{n-1} \cdot \Delta t \end{array} \right]}{\pi \cdot Jm \cdot (Ri \cdot \Delta t + Li)}$$

It needs the following algorithm to be implemented in the function Nr_incr_f, in order to compute the transient response.

$Nr_0 \leftarrow Nr\_init$

$Nr_1 \leftarrow Nr\_init$

"Computation loop"

for $n \in 2 .. n\_max$

$Nr_{n+1} \leftarrow Nr\_n\_plus\_un\_f(Udc_n, Nr_n, Nr_{n-1}, Cro_n, Cro_{n-1}, Jm, k\phi, Ri, Li, \Delta t)$

Nr

Comparison of the simulation methods

In order to be able to compare the different methods, let us define the voltage and torque successive values:

$Udc :=$ for $n \in 0 .. n\_max$, $Udc_n \leftarrow 50V$, Udc

$Cro :=$ for $n \in 0 .. n\_max$, $Cro_n \leftarrow 1N \cdot m$, Cro

$\Delta t := 10^{-3} s$

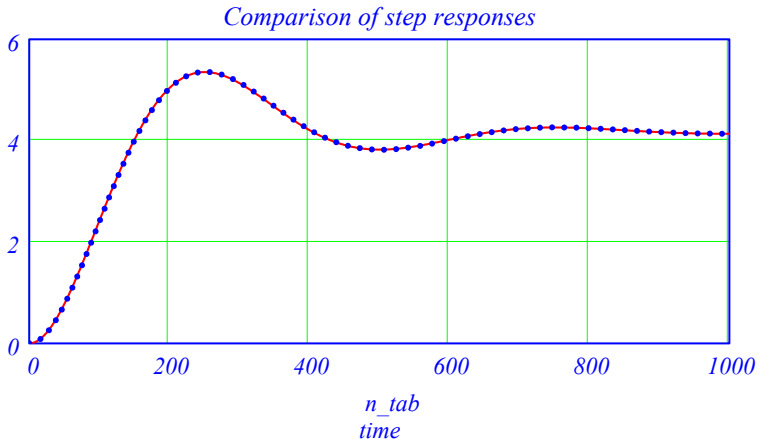
Let us define the numerical time steps $n\_tab := 0 .. n\_max$, and the corresponding Laplace and incremental functions:

$Nr\_Laplace(to, Udco, Cro, Ri, Jm) := Nr\_Laplace\_f(to, Udco, Cro, Ri, Li, k\phi, Jm, 0A, 0Hz)$

$Nr\_incr\_f(Cro, Udc, Ri, Jm) := Nr\_incr\_f(Cro, Udc, Nr\_init, n\_max, k\phi, Ri, Li, Jm, \Delta t)$

speed $Nr\_Laplace(n\_tab \cdot \Delta t, Udc_0, Cro_0, 0.1 \cdot \Omega, 2000 \cdot gm \cdot m^2)$

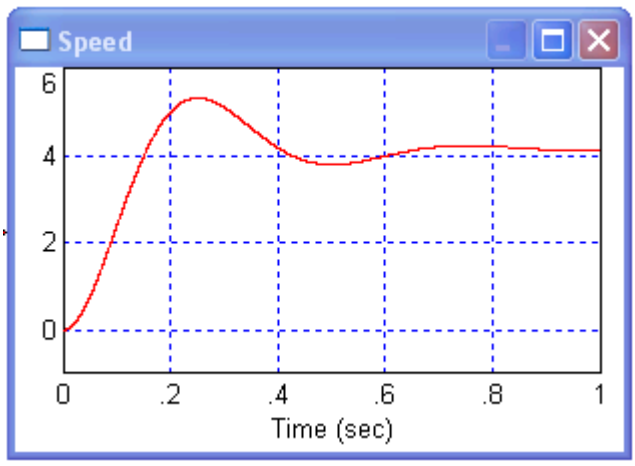
$Nr\_incr\_f(Cro, Udc, 0.1\Omega, 2000gm \cdot m^2)_{n\_tab}$



VisSim transient model

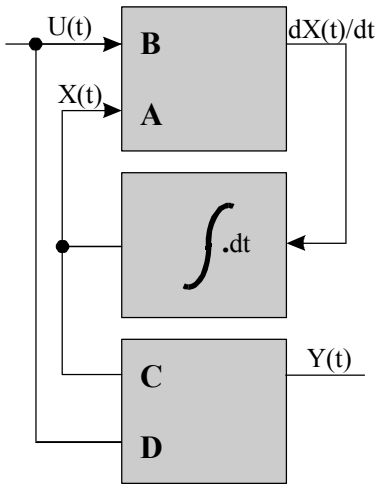
It is a good news, descovering that both methods give the same result for step response. It is better to realize that **VisSim** gives exactly the same result, for less development time.

The response given at left, is obtained with the VisSim diagram: *Mdc PM state space.vsm*



Are you now conviced that you should download **VisSim** on your computer (full free demo at http://www.vissim.com/downloads/demos.html)?

Construction of the motor VisSim dynamic model



The best way to construct a simulation model consists of utilizing state space variables, in the standard form:

$$\frac{d}{dt}X(t) = A \cdot X(t) + B \cdot U(t)$$

$$Y(t) = C \cdot X(t) + D \cdot U(t)$$

In our case, the motor model gives us:

$$Udc(t) = Ei(t) + Ri \cdot Ii(t) + Li \cdot \frac{d}{dt}Ii(t)$$

$$Ta(t) = Jm \cdot \frac{d}{dt}2\pi Nr(t) + Cro(t)$$

and therefore

$$Udc(t) = k\phi \cdot 2 \cdot \pi \cdot Nr(t) + Ri \cdot \frac{Ta(t)}{k\phi} + Li \cdot \frac{d}{dt}\frac{Ta(t)}{k\phi}$$

$$Ta(t) = Jm \cdot \frac{d}{dt}2\pi Nr(t) + Cro(t)$$

The state space model is given by the following matrix equations:

$$\frac{d}{dt}\begin{pmatrix} Ta(t) \\ Nr(t) \end{pmatrix} = \begin{pmatrix} \frac{-Ri}{Li} & \frac{-2 \cdot \pi \cdot k\phi^2}{Li} \\ \frac{1}{2 \cdot \pi \cdot Jm} & 0 \end{pmatrix} \cdot \begin{pmatrix} Ta(t) \\ Nr(t) \end{pmatrix} + \begin{pmatrix} \frac{k\phi}{Li} & 0 \\ 0 & \frac{-1}{2 \cdot \pi \cdot Jm} \end{pmatrix} \cdot \begin{pmatrix} Udc(t) \\ Cro(t) \end{pmatrix}$$

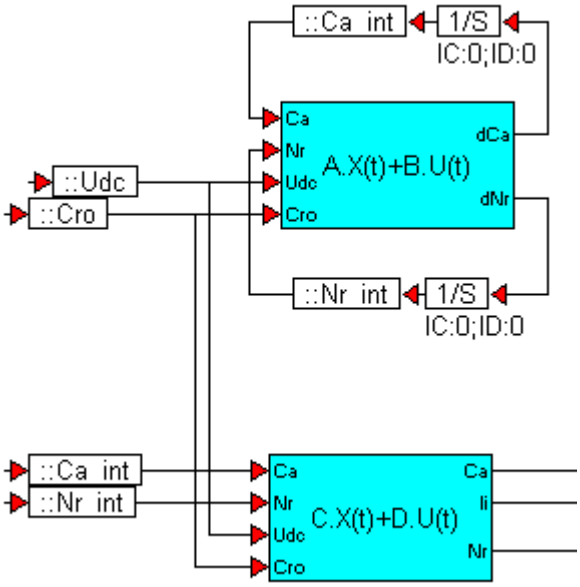
$$\begin{pmatrix} Ta(t) \\ Ii(t) \\ Nr(t) \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \frac{1}{k\phi} & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} Ta(t) \\ Nr(t) \end{pmatrix} + 0 \cdot \begin{pmatrix} Udc(t) \\ Cro(t) \end{pmatrix}$$

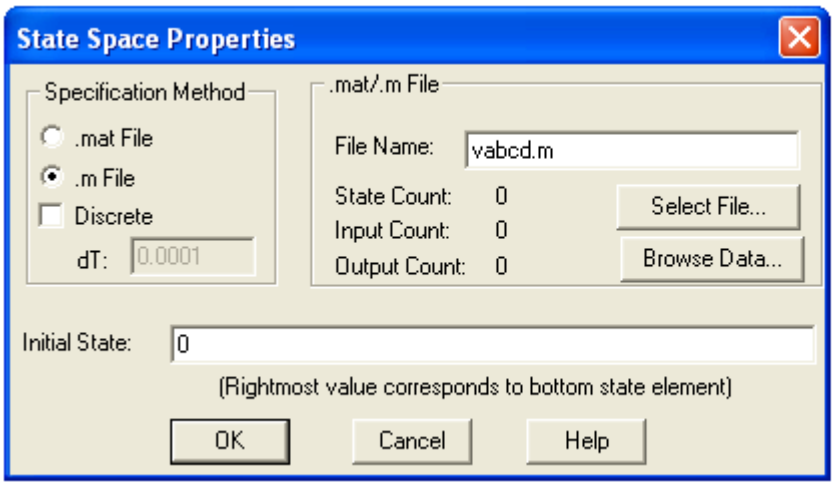
VisSim allows to build state space diagrams (see right) or tu use embeded ones (see below).

It is possible to embeded true non-linearities, as thermal estimator linked to the rotor resistance, saturation effects with an influence on the rotor coefficient $k\Phi$..

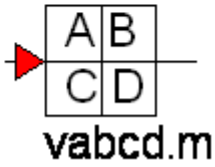
The full **VisSim** diagram is given with this demo (*Mdc PM state space.vsm*).

VisSim self made State Space Variables block





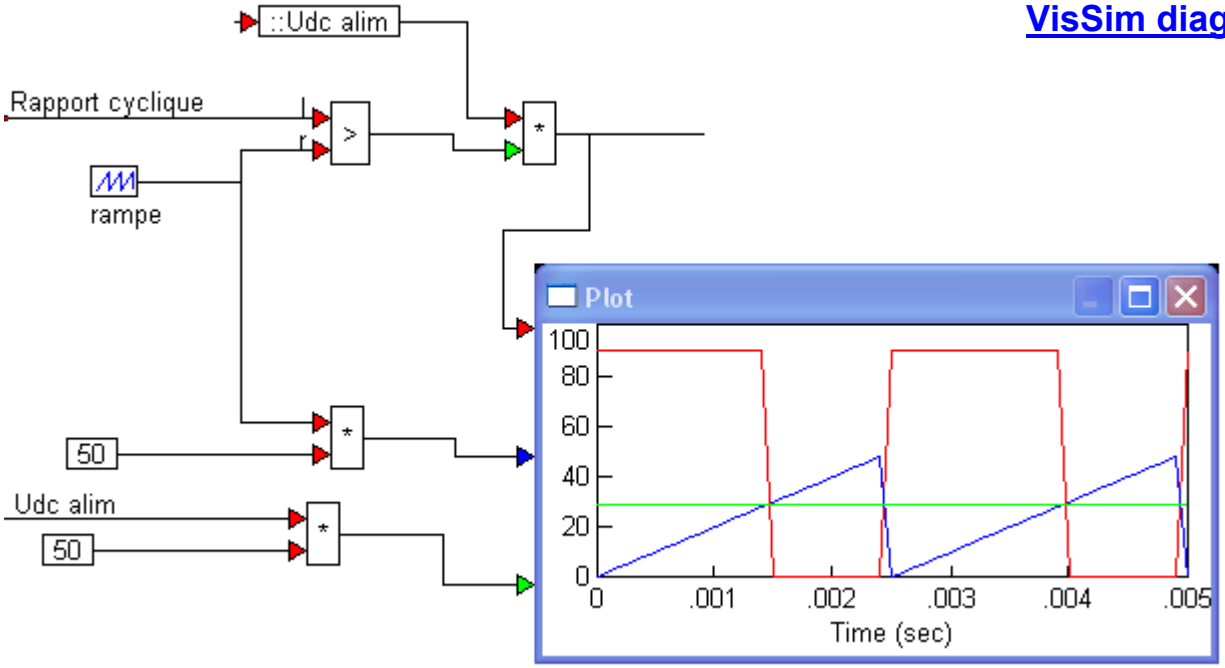
VisSim
State Space Variables
block



Simulation of the chopper

It could be possible to simulate the chopper with **Mathcad,** I did it for other problems, but it much more efficient to do it with **VisSim.** Next figure gives a simple an effective solution.

VisSim diagram



Conclusion: VisSim is the design chain solution

I tried to give in this document a personal testimony of the complementarity between **Mathcad** and **VisSim**. **Mathcad** is a very good tool to prepare the work sheet that **VisSim** will use. It allows to describe clearly all the problem and to transmit the knowledge to people who will go further in the work.

VisSim with its full real time computation ability allows to go much further than **Mathcad**. If I could convince you to test **VisSim**, then let open the two given diagrams: *Mdc PM state space.vsm* & *Mdc PM integer + Chopper with Torque & Speed loop simulated PID.vsm*. They will give you a very good example of how clear and efficient are **VisSim** diagrams. Make sure that the given images are put in the same folder as **VisSim** files, if you want to enjoy the images in the blocks.

But **VisSim** allows you to go further. Indeed, it can drive DSP board, with full ability. With such a **VisSim** design chain, the engineer can process all the study steps of an engineer problem:

- design step with **Mathcad** and **VisSim**
- simulation process with **VisSim** and feedback with the design teams
- embeding in DSP controled boards with real time control from the computer with **VisSim** embeded tools
- final product stand alone, with **VisSim** compiled download ability

Next image represents a power inverter of my design, dedicated to **VisSim** control. It receives a DSP board. It receives D/A control inputs and resolver input. It is compatible with F2407, F2812 and F2808 DSP boards from Texas Instruments. It can drive MS brushless, MAS & Mdc brush motors, and other loads.

VisSim DSP controlled inverter board

Units $rpm \equiv \frac{Hz}{60}$

Motor datas

Rotor resistance $Ri \equiv 0.1 \cdot \Omega$
Rotor inductance $Li \equiv 10 mH$
Rotor unloaded nominal voltage $Ein \equiv 200 V$
Rotor nominal current $Iin \equiv 10 A$
Nominal rotor speed $Nrn \equiv 1000 rpm$

Chopper datas

dc voltage source $Vdc \equiv 200 V$
Chopping frequency $fdec \equiv 10 kHz$

Incremental simulation datas

- $n\_max \equiv 1000$
- $Nr\_init \equiv 0 Hz$
- $Jm \equiv 10 gm \cdot m^2$