Olivier Colin

Going dynamics...

Using static results from Quickfield for fast and accurate simulations of actuators in Saber

Going dynamics Saber seminar 2013

- Generalities - Modelling magnetical aspects
- Principles - Simplifying equations
- Quickfield - A fast and versatile 2D software
- Tools - Automated Model Creation
- Applications - Linear actuator - 3-phases motor

Going dynamics - Generalities (1/2)

The need of using the same simulation tool becomes evident when different technologies require to be simulated in the same design, preferably a 1D tool (using schematics and discrete components such as Saber) , the reasons are :

The possibility to simulate large systems

Fast simulation and good accuracy

The Cost of ownership

But component techniques modelling is **not straightforward**

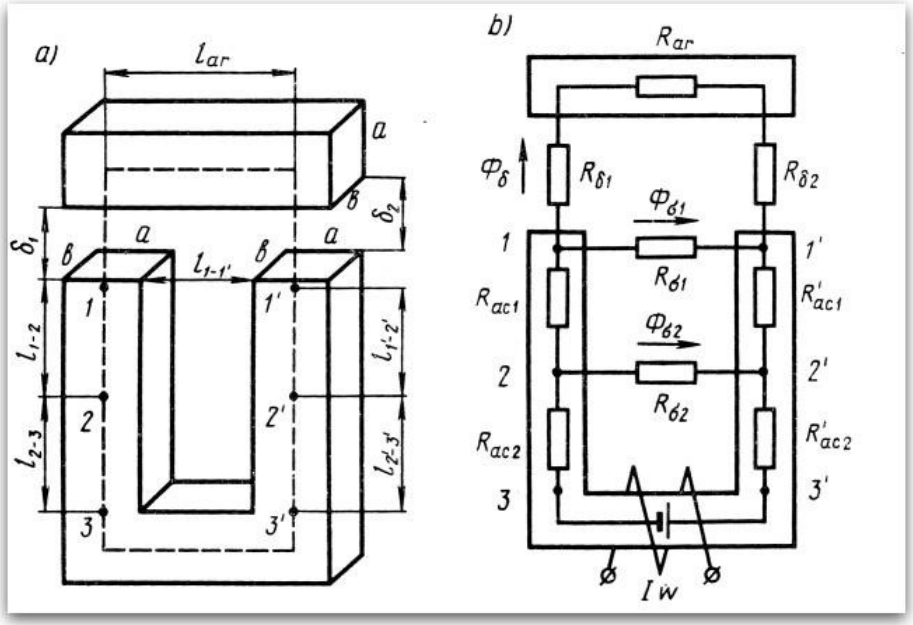
Going dynamics - Generalities (2/2)

Case of electromagnetics :

To create a model, we need to translate :

- Geometry
- Material caracteristics
- Physical interactions
- Movements

in a 1D description



1D representation of a magnetic circuit

Going dynamics - Principles (1 / 3)

Mechanical aspects

$$\frac{\partial(\text{mass}\cdot\text{speed})}{\partial(t)} = \sum(\text{Forces})$$

$$\frac{\partial(J\cdot\text{speed})}{\partial(t)} = \sum(\text{Torques})$$

Let's have a 2D table :

Forces	F(I,translation)
Torques	T(I,rotation)

Electrical aspects

$$\text{emf} = \frac{\partial(\Phi)}{dt}$$

Let's have a 2D table :

Φ	Flux(I, translation)
Φ	Flux(I, rotation)

Going dynamics - Principles (2 / 3)

For instance :

Mech Table :=

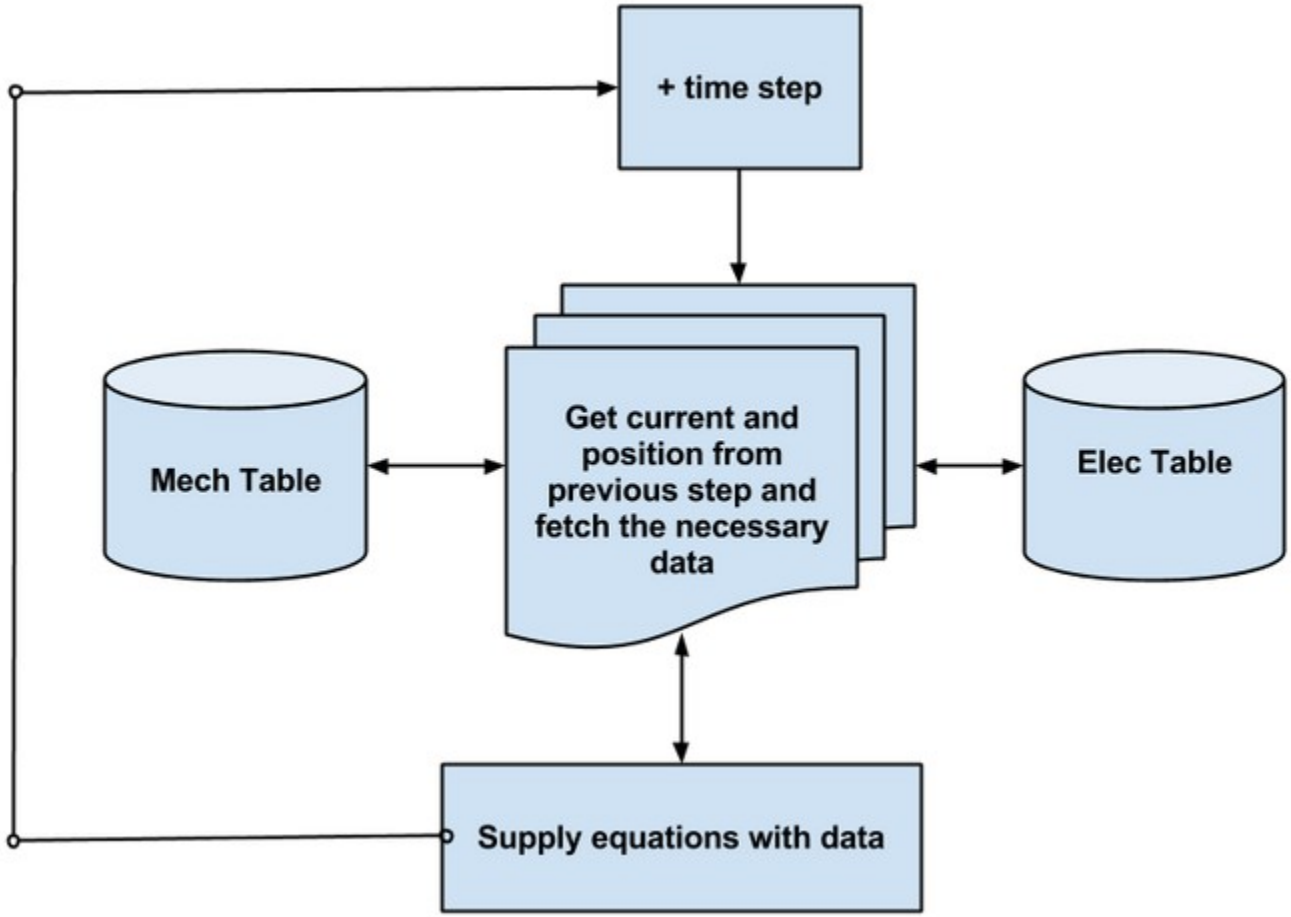
x	−I3	−I2	−I1	0	I1	I2	I3
x1	mF13	mF12	mF11	mF10	F11	F12	F13
x2	mF23	mF22	mF21	mF20	F21	F22	F23
x3	mF33	mF32	mF31	mF30	F31	F32	F33

Elec Table :=

x	−I3	−I2	−I1	0	I1	I2	I3
x1	mΦ13	mΦ12	mΦ11	mΦ10	Φ11	Φ12	Φ13
x2	mΦ23	mΦ22	mΦ21	mΦ20	Φ21	Φ22	Φ23
x3	mΦ33	mΦ32	mΦ31	mΦ30	Φ31	Φ32	Φ33

Going dynamics - Principles (3 / 3)

The algorithm is quite simple :



Going dynamics - QuickField (1/3)

QuickField is a Finite Element Analysis software with the following options

- Magnetic Suite
- DC Magnetics, AC Magnetics, Transient + DC Magnetics

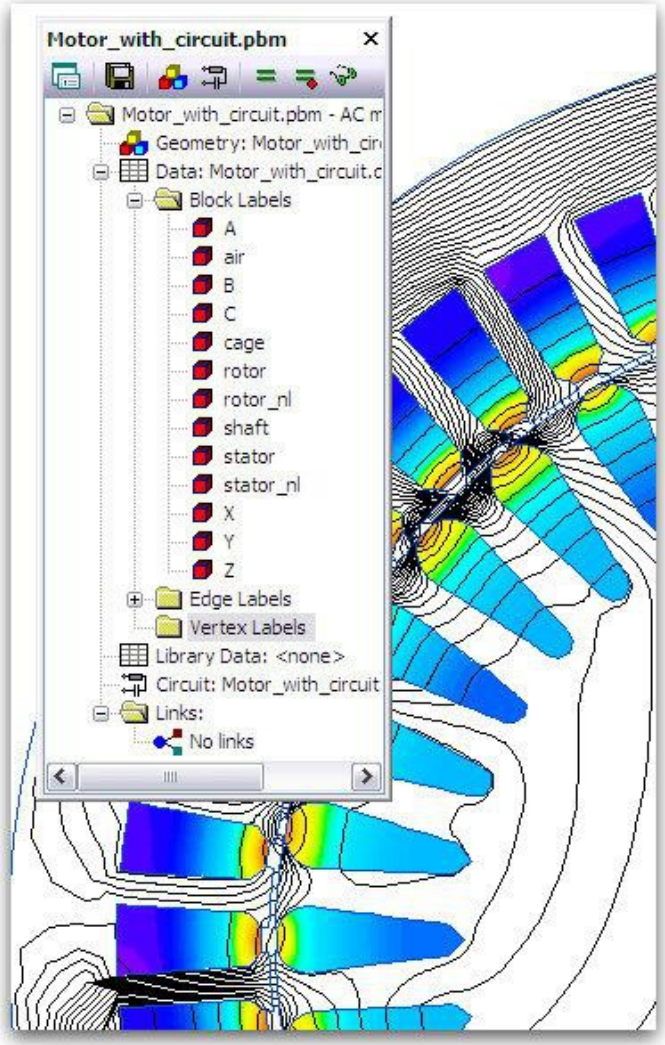
- Electric Suite
- Electrostatics & DC conduction, AC conduction, Transient Electric + Electrostatics & DC conduction

- Thermostructural Suite
  Stress Analysis, Steady and Transient Heat transfer

Going dynamics - QuickField (2/3)

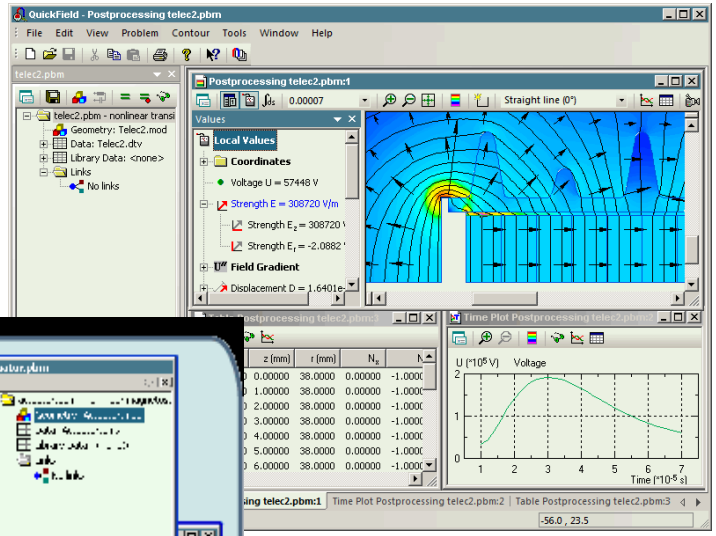
- Fast FEA software
- Easy automation

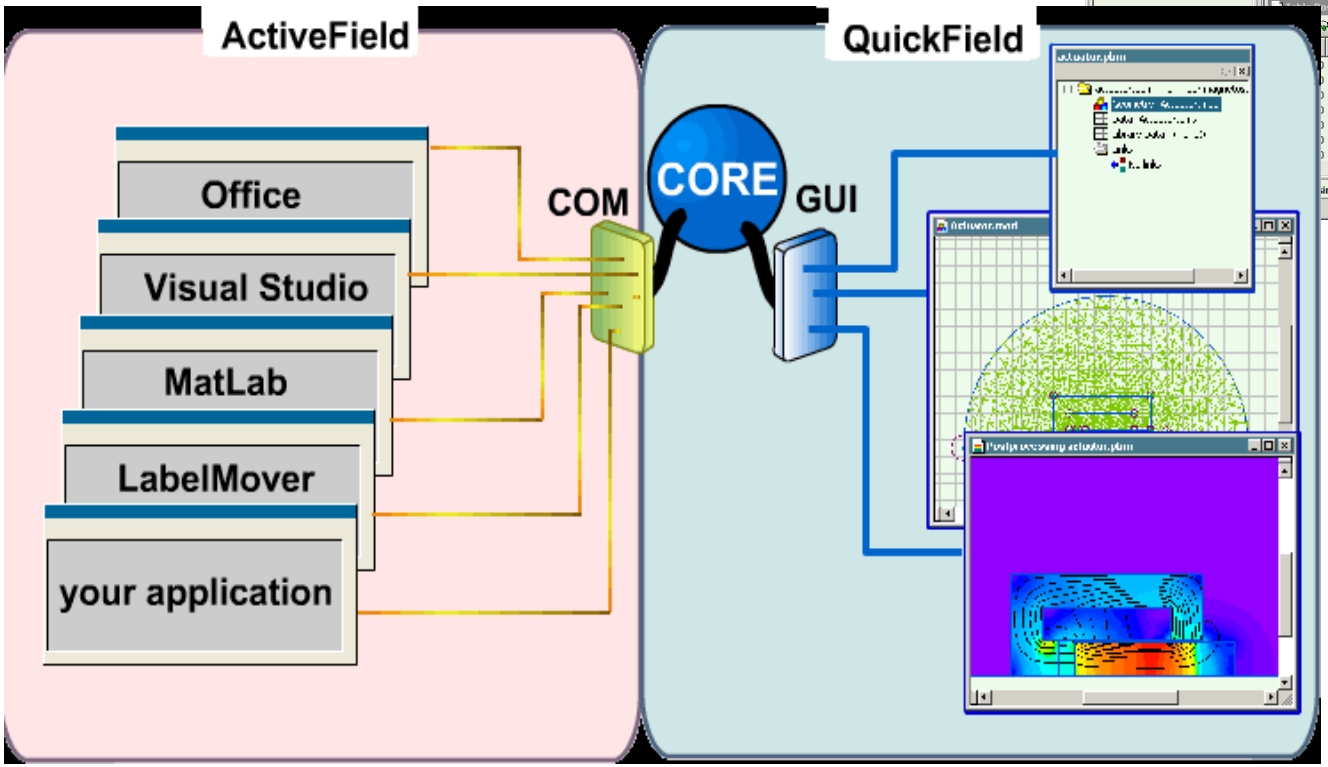
The design is changed step by step. the simulation of static operating points is easier to realize than a dynamic simulation. At each step, the magnetical flux and mechanical forces are stored.



Going dynamics - QuickField (3/3)

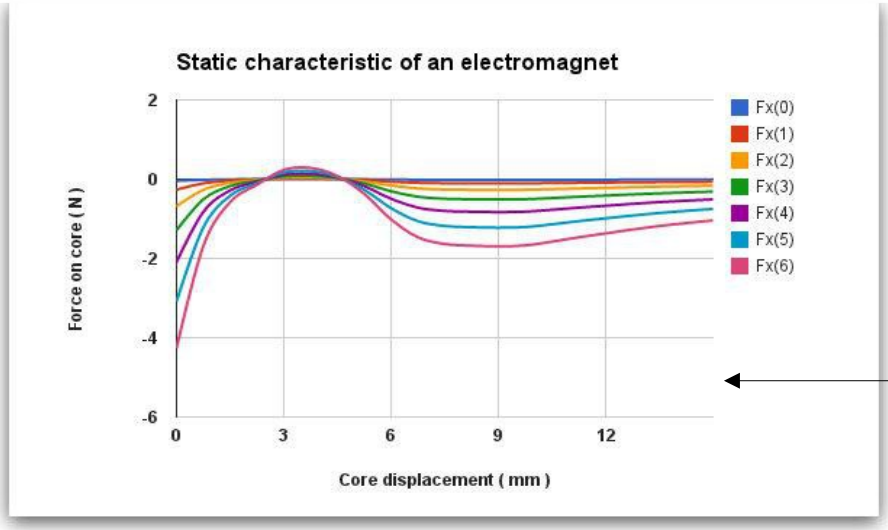
Automation in QuickField : ActiveField



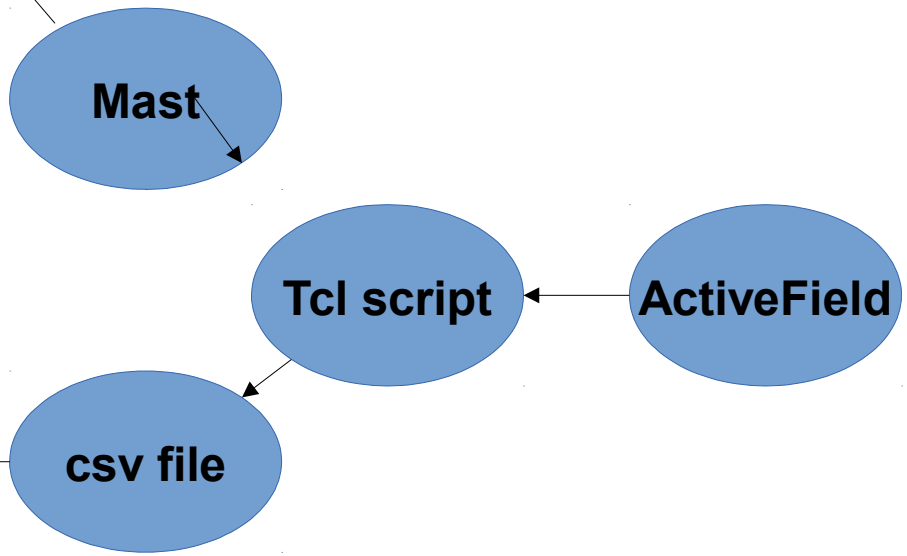


Going dynamics - Tools

```
foreign pwl1

parameters {

    force_0 = [(0,0.000000),(0.00075,0.000000),
    force_1 = [(0,-0.118585),(0.00075,-0.026270
    force_2 = [(0,-0.474339),(0.00075,-0.105075
    force_3 = [(0,-1.067259),(0.00075,-0.236417
    force_4 = [(0,-1.897338),(0.00075,-0.420292
    force_5 = [(0,-2.964582),(0.00075,-0.656696
    force_6 = [(0,-4.268973),(0.00075,-0.945634
    force_7 = [(0,-35.890573),(0.00075,-7.95019
```



The Tcl script change the geometry and fill a tabl



Going dynamics - Mast (1 / 4)

Mast declaration

```
val frc_N f_0 ,f_1 ,f_2 ,f_3 ,f_4, f_5, f_6, f_7
val frc_N f_n0 ,f_n1 ,f_n2 ,f_n3 ,f_n4, f_n5, f_n6, f_n7
val f fl_0 ,fl_1 ,fl_2 ,fl_3 ,fl_4, fl_5, fl_6, fl_7
val f fl_n0 ,fl_n1 ,fl_n2 ,fl_n3 ,fl_n4, fl_n5, fl_n6, fl_

⟶
struc { number xi, yi ; }  \
⟶force_0[*], force_1[*], force_2[*], force_3[*], force_
⟶force_5[*], force_6[*], force_7[*], \
⟶flux_0[*], flux_1[*], flux_2[*], flux_3[*], flux_4[*]
⟶flux_5[*], flux_6[*], flux_7[*]
struc { number xi, yi ; }  \
⟶force_n0[*], force_n1[*], force_n2[*], force_n3[*], f
⟶force_n5[*], force_n6[*], force_n7[*], \
⟶flux_n0[*], flux_n1[*], flux_n2[*], flux_n3[*], flux_
⟶flux_n5[*], flux_n6[*], flux_n7[*]


number idens_0, idens_1, idens_2, idens_3, idens_4, idens_
number idens_n0, idens_n1, idens_n2, idens_n3, idens_n4, i
```

Going dynamics - Mast (2 / 4)

parameters section

```
foreign pwl1

parameters {

    force_0 = [(0,0.000000),(0.00075,0.000000),
    force_1 = [(0,-0.118585),(0.00075,-0.026270
    force_2 = [(0,-0.474339),(0.00075,-0.105075
    force_3 = [(0,-1.067259),(0.00075,-0.236417
```

values section

```
values {

#---+----1----+----2----+----3----+----4----+----5-
# Force
#---+----1----+----2----+----3----+----4----+----5-

    # the positive part

    f_0 = pwl1(12, addr(force_0), len(force_0), x)
    f_1 = pwl1(12, addr(force_1), len(force_1), x)
    f_2 = pwl1(12, addr(force_2), len(force_2), x)
```

Going dynamics - Mast (3 / 4)

Values section (cont.)

```
#---+----1----+----2----+----3----+----4----+----5---
# All
#---+----1----+----2----+----3----+----4----+----5---

    if (idens >= idens_0 & idens < idens_1) {
        force = ( f_1 - f_0 ) / (idens_1 - idens_0)*(
        flux = ( fl_1 - fl_0 ) / (idens_1 - idens_0)*
    }

    else if ( idens >= idens_1 & idens < idens_2 ) {
        force = ( f_2 - f_1 ) / (idens_2 - idens_1)*(
        flux = ( fl_2 - fl_1 ) / (idens_2 - idens_1)*
    }
```

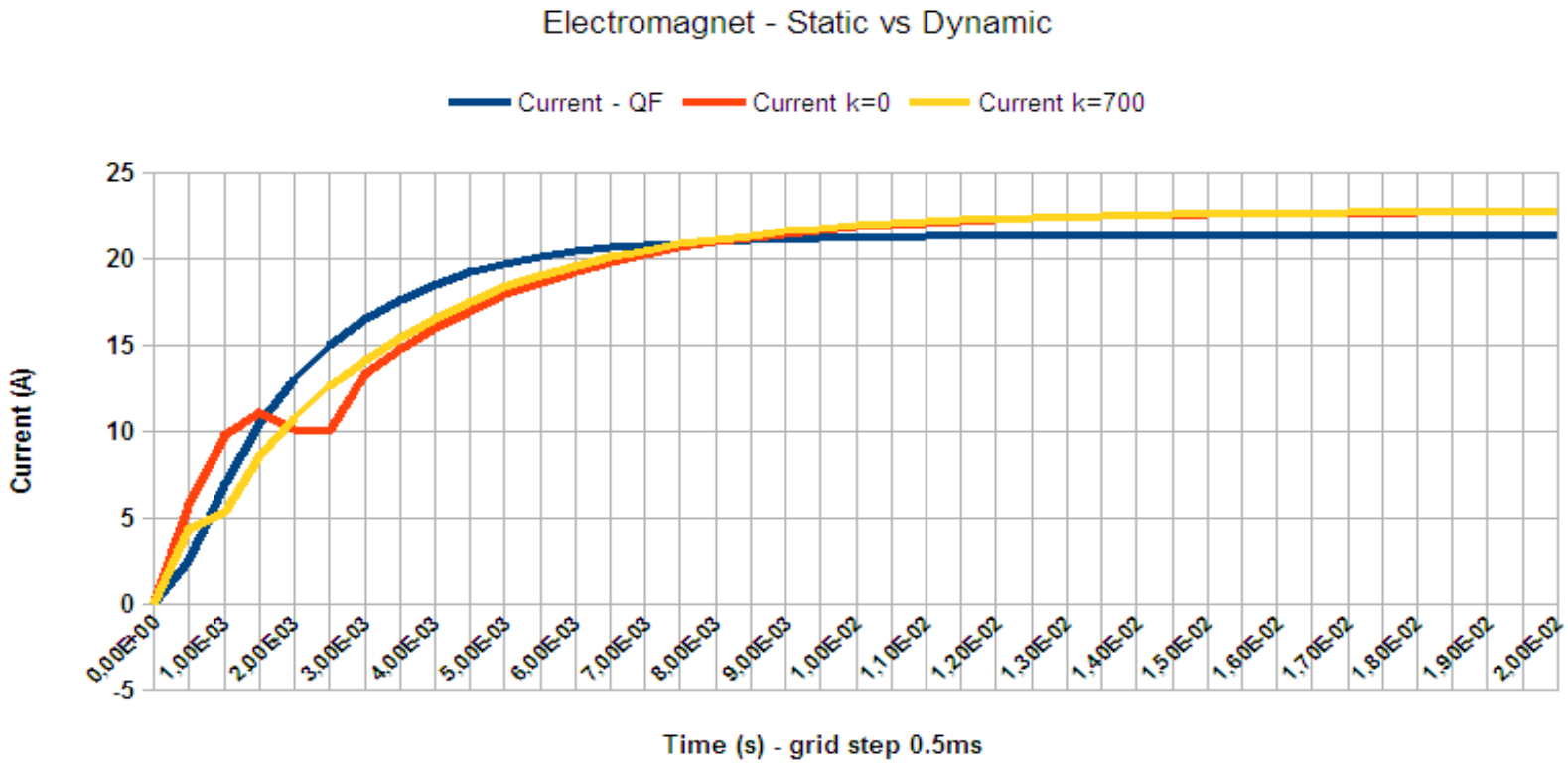
Going dynamics - Mast (4 / 4)

control and equations section

```
control_section {

    # newton_step(idens,ns_idens)
    # sample_points(x,sp_x)
    initial_condition(x,13m)
    initial_condition(speed,0)
}

equations {

    i(p) += i
    i(m) -= i
    fcem : fcem = d_by_dt(N*flux)
    i : v = fcem + vr
    x : d_by_dt(x) = speed
    speed : d_by_dt(speed)=(force-k*x)/mass
```

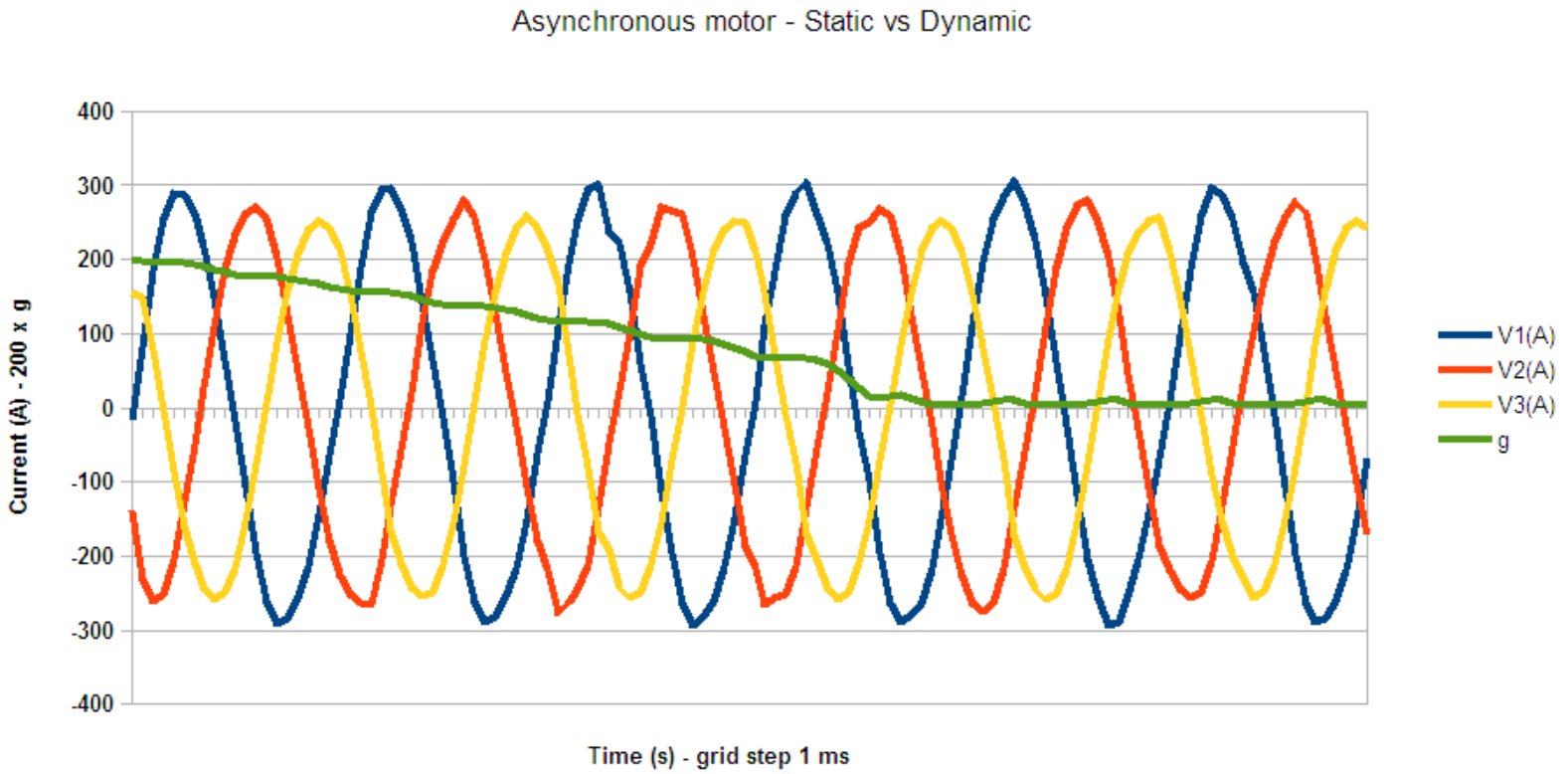
Going dynamics - Applications (1/2)

Linear actuators



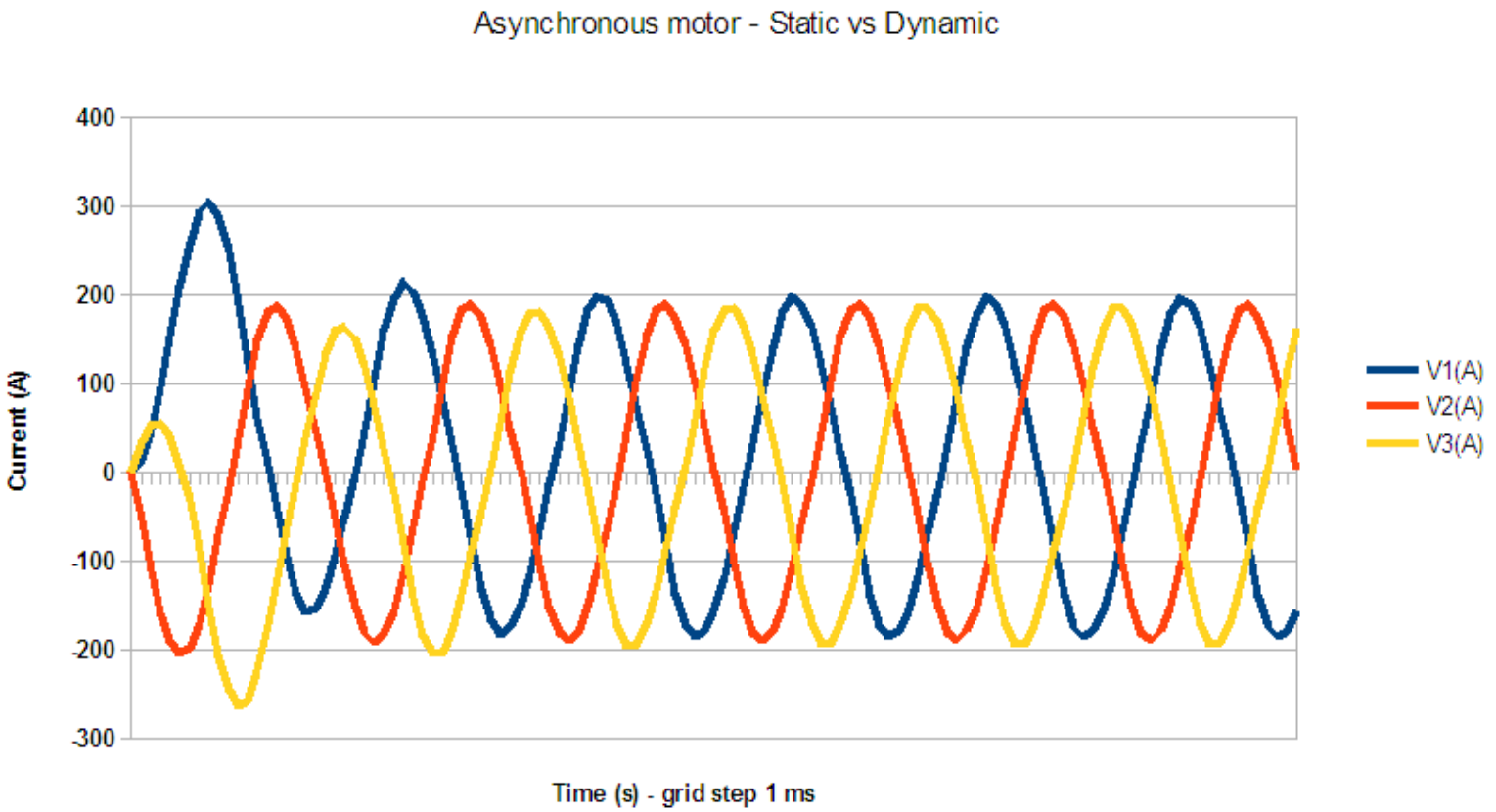
Going dynamics - Applications (2a/2)

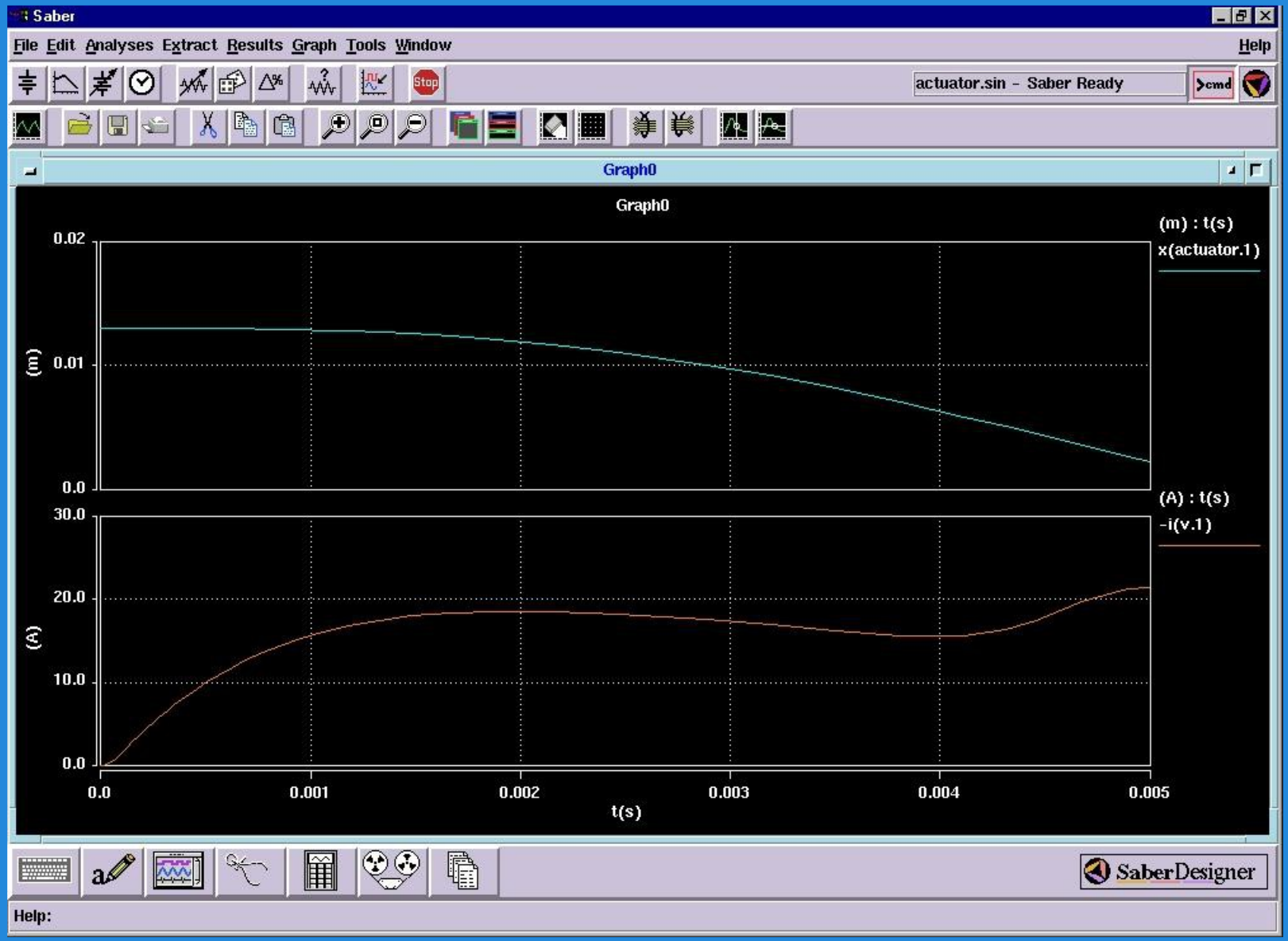
Rotary actuators



Going dynamics - Applications (2b/2)

Rotary actuators





Saber will stay a unique tool, powerful, simple to use and that, with only about thirty Mast words to know….