



End to End model inside out

Hiro Yamamoto / CIT



- ◆ e2e simulation package
 - » GUI
 - » Simulation Engine
 - » Helper apps
- ◆ Time domain simulation
- ◆ Mechanics simulations
 - » base model
 - » simple 3d mirrors
 - » Mechanical simulation engine
- ◆ Things to do

- * Optics simulation Biplab
- * Software distribution and maintenance Ed

2:00 PM

- * Lock Acquisition Matt
- * Misalignment effect Luca
- * In-lock state noise Biplab

Oct. 19, 20

- * In depth discussion of e2e
- * How to support e2e at LHO&LLO
- * Future collaboration



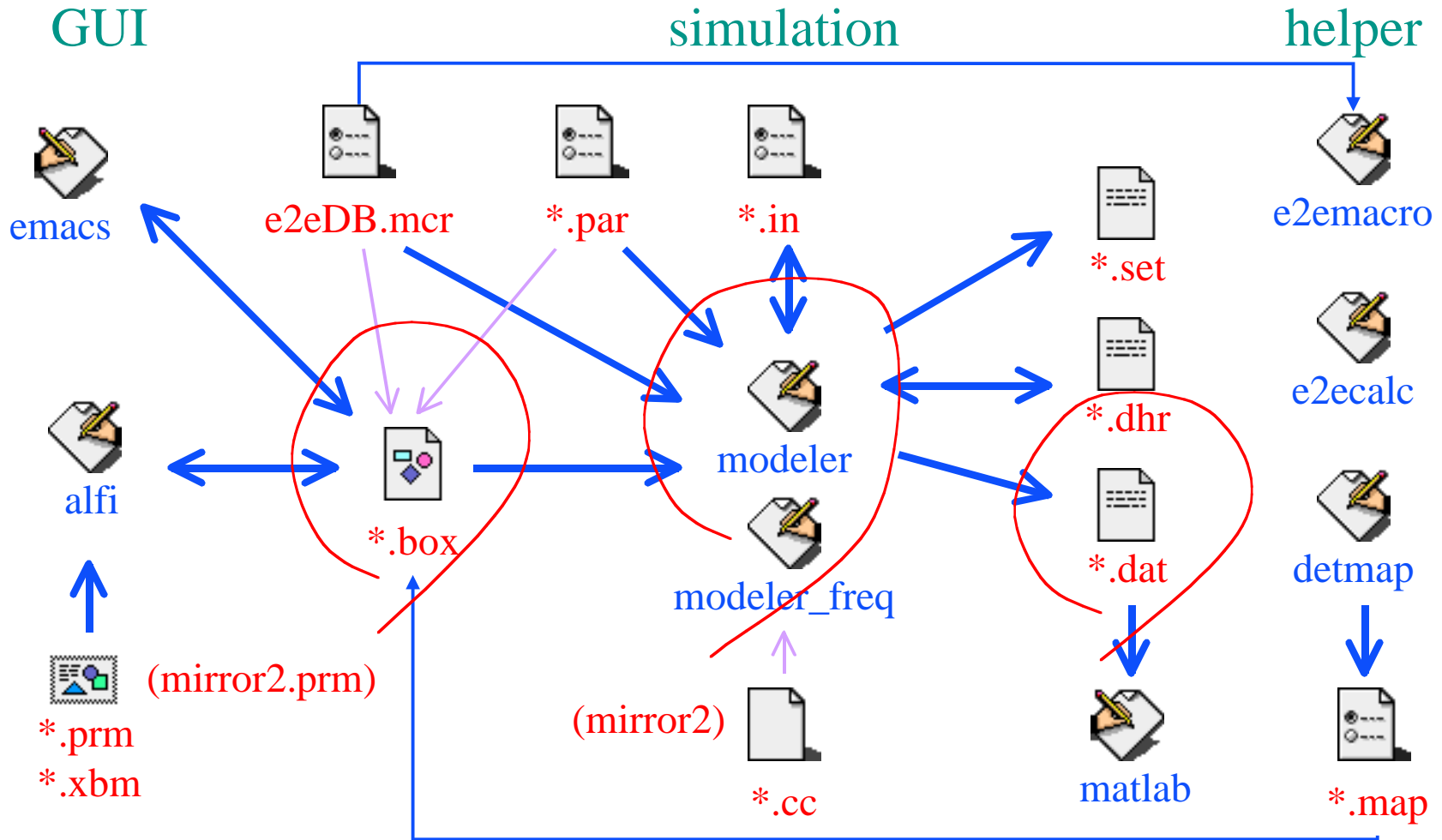
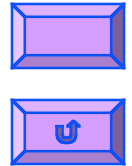
End to End simulation what is it



- ◆ General purpose GW interferometer simulation program
 - » Generic tool like matlab or mathematica
 - » Easy to simulate a wide range of configuration without modifying and revalidating codes
- ◆ Simulation program
 - » Time domain simulation written in C++
 - » Optics, mechanics, servo ...
 - » Easy to add new phenomena by concentrating on physics, not on programming



E2E simulation package Components





Box files

Simulation setup



Alfi (GUI) view

Parameter ✓
 Value
 Type real

	Parameter	Value
1	max_mode_order	2
2	waist_size_X	0.0008
3	waist_size_Y	0.0008
4	distance_waist_X	distz
5	distance_waist_Y	distz
6	angle_resolution	1e-10
7	compute_mismatch_curvature	yes
8	power	1

data_in.xbm (points to Power component)
 field_gen.prm (points to Laser component)

Content of box file

```

Add_Submodules
{
  field_gen Laser;
  mirror2 ITM; mirror2 ETM;
  propagator I2E; propagator E2I;
  data_in Power; ...
}

Settings Laser
{
  max_mode_order = 2
  distance_waist_X = distz
  compute_mismatch_curvature = yes; ...
}

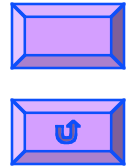
Settings I2E { length = CavLength }
Settings E2I { length = CavLength }
Settings Power {init = 1; type = vector_real}
...

Add_Connections
{
  Power 0 -> Laser power; Laser 0 -> ITM Bin;
  ITM Aout -> I2E 0; I2E 0 -> ETM Ain
  ETM Aout -> E2I 0; E2I 0 -> ITM Ain
  ETM Bout -> TransPower 0;
  TransPower 0 -> this Tran0 ...
}

```



Auxiliary inputs



e2eDB.mcr

- > macros defined here can be used anywhere in the simulation

```
<
% This is a macro db for Curv.box
>
CavLength = 2.655 [m] "arm length"
ditz      = 0.1   [m] "waist displacement"
E01       = 0.1   [W] "TEM01 mode power"
```

*.par

- > out most input port value
- > data_in values can be redefined here
- > data_in module name = value
- > They can change during a run

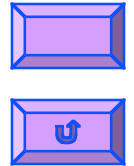
```
Power = sqrt(100-2*pow(E01,2)),E01,E01
```

macro definition as runtime option

- > -param name = value
- > e.g., modeler -param ditz = -0.1



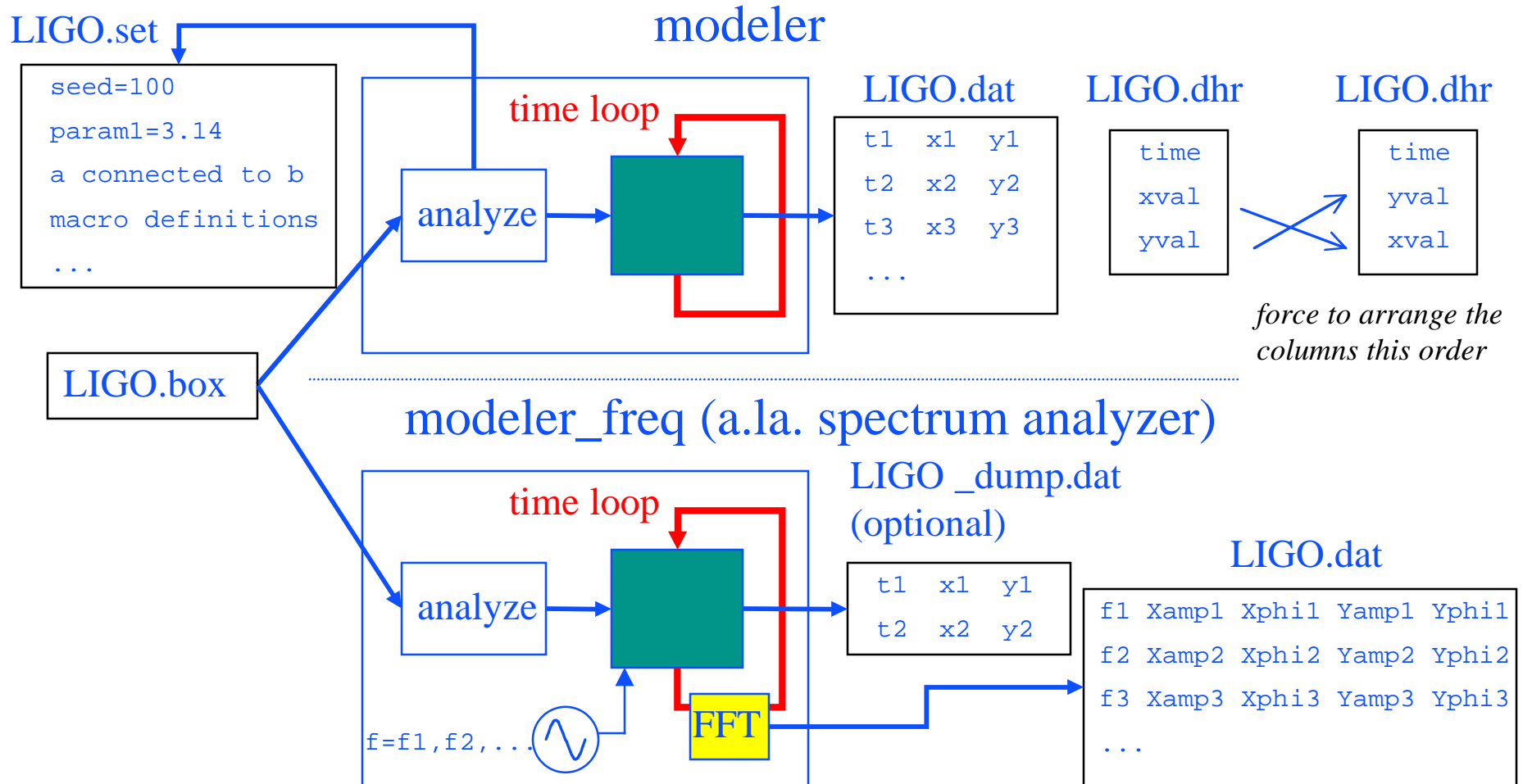
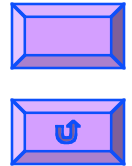
Simulation engine Core



- ◆ digitized time evolution ($10^{-3} \sim 10^{-7}$ sec, L/c)
- ◆ optics, mechanics, digital filter (no GW source yet)
- ◆ primitive modules
 - » simulating physical devices
 - mirror, laser, EOM, PD+mixer, PRM, single suspended mirror,...
 - » calculation tools
 - math functions, logic, vector operations,...
 - equation parser
 - | supplement shortcoming of iconic programming
 - | use c-like expressions with many math functions supported
- ◆ Based on the description in the input box files, these primitives are combined in a proper order to simulate the system.



Simulation engine framework





Simulation engine how to run



running modeler

```
Loading data from "./e2eDB.mcr"  
% This is a macro db for Curv.box  
E2E> Description file >> Curv.box  
E2E> Parameter file ('.' for none)  
(def="Curv.par") >> Curv.par  
E2E> Output File Name ('.' to halt):  
(def="Curv.dat") >> Curv.dat  
E2E> Setting File Name ('.' for no output):  
(def="Curv.set") >> Curv.set  
E2E> Model time step (def=1e-05,[0:INF]) >>  
8.856e-9  
Current time is 0.000 (s).  
E2E> Simulation time (s) (def=1,[0:INF]) >>  
1e-4  
This will require 11291 steps.  
E2E> Write one data point every N steps. (0  
to halt) (def=1,[0:INF]) >> 100  
Rereading parameter values...  
Running...  
Done Running.  
E2E> Continue? >> no  
Done.
```

chance to
edit par file

interaction is recordable

@(fname ~ @)

c.in

modeler < c.in

```
% Description file >>  
Curv.box  
% Parameter file ('.' for none) >>  
Curv.par  
% Output File Name ('.' to halt): >>  
Curv.dat  
% Setting File Name ('.' for no output): >>  
Curv.set  
% Model time step (def=1e-05,[0:INF]) >>  
8.856e-9  
% Simulation time (s) (def=1,[0:INF]) >>  
1e-4  
% Write one data point every N steps. >>  
100  
% Continue? >>  
no
```

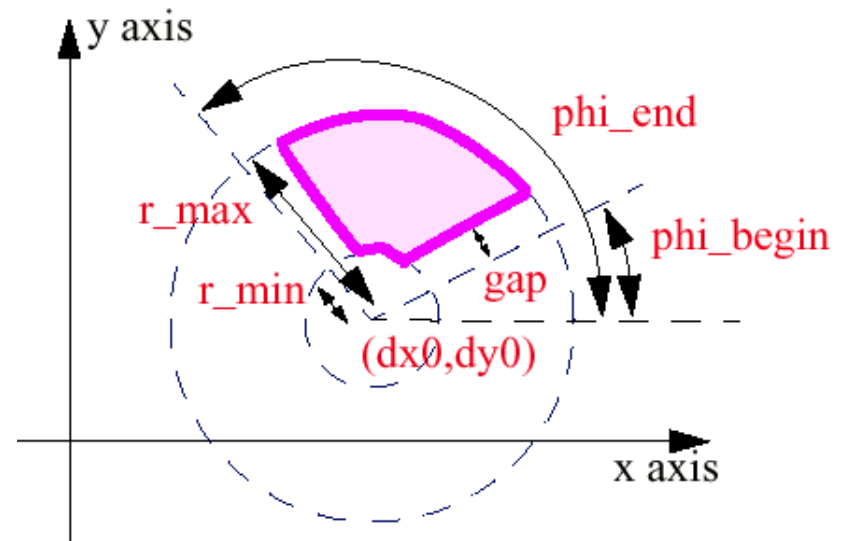
Helper applications



detmap : generates data file for photo detector
with arbitrary shape with non uniform efficiency

```

E2E> Define the detector shape (lengths are in
units of beam size on detector)
E2E>
E2E>   r_min      [0:INF]      = 0
E2E>   r_max      [0:INF]      = 5
E2E>   phi_begin  [-180:360]   = 0
E2E>   phi_end    [0:INF]      = 360
E2E>   gap        [0:INF]      = 0
E2E>   dx0        [0:INF]      = 0
E2E>   dy0        [0:INF]      = 0
E2E>
E2E> "name = val", "?" for help or OK >> ok
E2E> max of m+n (def=3,[0:INF]) >> 3
E2E> Do you want to define non uniformity ?
E2E>     none      *one line def   read file
E2E> Type "name" to toggle on/off or OK >> ok
E2E> Equation >> 1-0.1*(x*x+y*y)
  
```

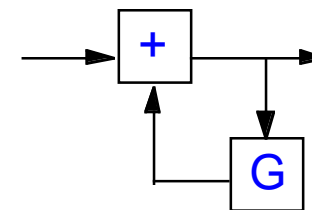
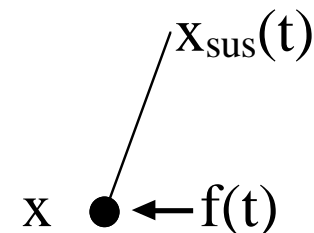


Time domain simulation



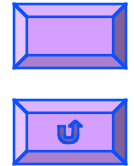
- ◆ Analog process is simulated by a discretized process with a very small time step ($10^{-7} \sim 10^{-3}$ s)
- ◆ Linear system response is handled using digital filter
 - » e2e DF = PF's pziir.m (bilinear trans (s->z) + SOS) + CDS filter.c
 - » Transfer function -> digital filter
 - » Pendulum motion
 - » Analog electronics
- ◆ Easy to include non linear effect
 - » Saturation, e.g.
- ◆ A loop should have a delay
 - » Need to put explicit delay when needed
 - » Need to choose small enough time step

$$x = \frac{1}{s^2 + \gamma s + \omega_0^2} \left(\frac{f}{m} + \omega_0^2 x_{sus} \right)$$

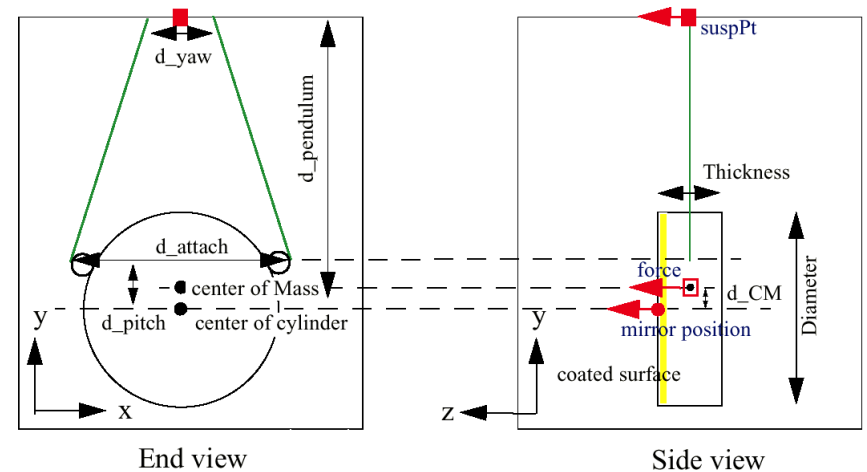
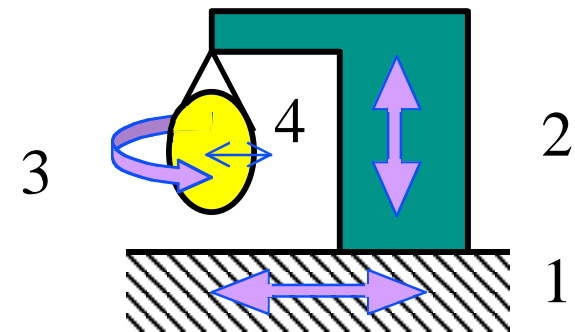




Mechanics simulation base model



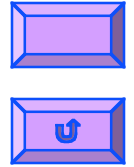
- (1) Seismic motion from measurement
- (2) Parametrized HYTEC stack
 - » Ed Daw
- (3) Simple single suspended mirror
 - » Malik
 - » 4 sensors and actuators
 - » couple between LSC and ASC
- (4) Thermal noise added in an ad hoc way
 - » using Sam Finn's model
 - » later by Biplab





Mechanical simulation

Giancarlo Cella of Pisa Univ.

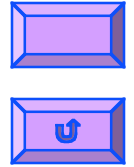


- ◆ C++ mechanics class library
 - » Mechanical Simulation Engine (MSE)
- ◆ Modular design with flexibility
 - » Simulation of wide varieties of mechanical structures
- ◆ Time domain and frequency domain simulation
- ◆ Easy integration with the optics and control parts of the End to End model
 - » Performance of the mechanics subsystem alone or
 - » Performance of the integrated system
- ◆ API for e2e has been completed
 - » Final integration has not been completed



MSE

general principle



- ◆ It is a fully three dimensional simulation.
 - » It is possible to calculate cross couplings of various degrees of freedom.
- ◆ It is a modular environment.
 - » The system is partitioned in subunities, and each of them can be modeled internally in an arbitrary way.
- ◆ A model can be selectively refined.
 - » Set the number of internal modes of a given mechanical component
 - » Choose a different representation for a component
- ◆ The equilibrium point for the system is automatically calculated
 - » Only the connections between elements

MSE & e2e PSL noise study

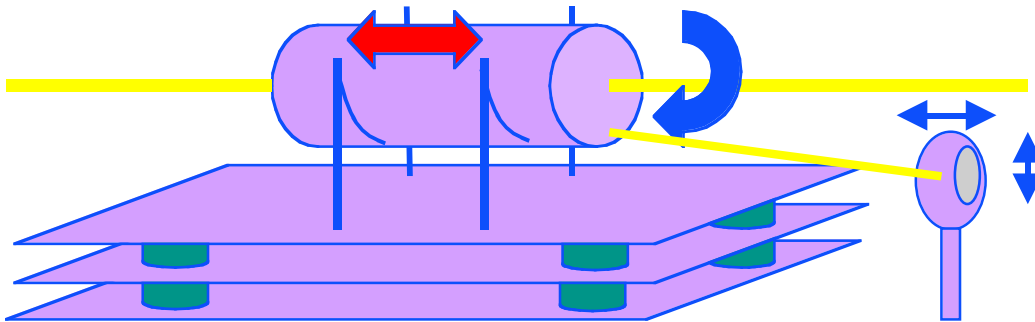
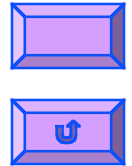
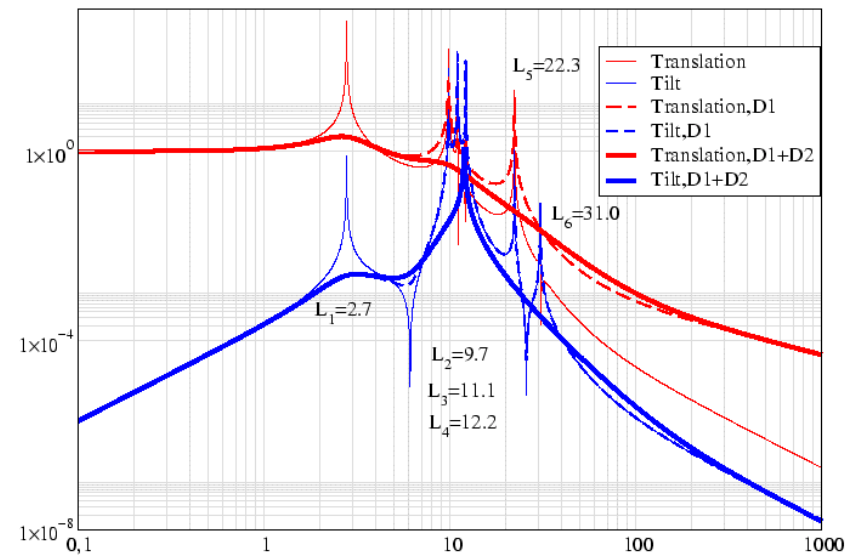


table motion to ref.
cavity motion
(displacement and tilt)

- ◆ PSL reference cavity model built in a day
- ◆ Cavity motion by table top motion + transfer function
- ◆ Noise study using cavity motion as input to e2e

X motion





Things to do



- ◆ Installation and maintenance of e2e at each site
- ◆ e2e experts at each site
- ◆ Site specific program (box files) development
- ◆ Sharing of programs
- ◆ Improvement of simulation code
 - » optics - thermal lensing, thermo-elastic, dual recycling, ...
 - » mechanics - MSE, more realistic parameters, ...
 - » digital filter - time step limitation
 - » speed - code improvements, thread, ...