

Real Controls with a Simulated Plant at the 40m

LIGO Caltech

Joe Betzwieser

Rana Adhikari, Koji Arai, Jameson Rollins

LIGO - Caltech / April 26, 2011

Why Simulate?

- To help **design**:
 - Combine many simple laws to produce complexity
 - Make sure we don't miss something due to that complexity
 - Make sure a design is plausible
- To help **understand**:
 - Difference between reality and simulation lies new understanding
 - Toy models can help suggest methods of control

Previous Simulations of Interferometers

- **E2E: End to End**
 - Full **Time domain** simulation of the fields
 - Simulates optics, mechanics, servos, etc
 - Written in **C++**
 - Outputs **text files** with fields and other info

- **Optickle** and its derivatives:
 - **Frequency domain** simulation of the fields
 - Written in **MATLAB**
 - Outputs **plots and text data**

- **GWINC:**
 - Static simulation of **noise spectra**
 - Written in **MATLAB**
 - Outputs **plots and text data**

Other Simulations of Interferometers

- Siesta (time domain, c, modal model, Virgo-Annecy/France)
- Twiddle(frequency domain, Mathematica, scalar field, LIGO)
- Finesse(frequency domain, c, modal model, GEO)
- MIST (frequency domain, matlab, modal model, Virgo-Pisa/Italy)
- SIS (stationary field, C++, FFT, aLIGO)
- MITFFT(static field, fortran, FFT, iLIGO)
- DarkF (static field, fortran, FFT, Virgo-Nice/France)
- Siesta (static field, c, FFT, Virgo-LMA/France) use tools of initial time domain siesta
- OSCAR(static field, matlab, FFT, Virgo-LMA/France)

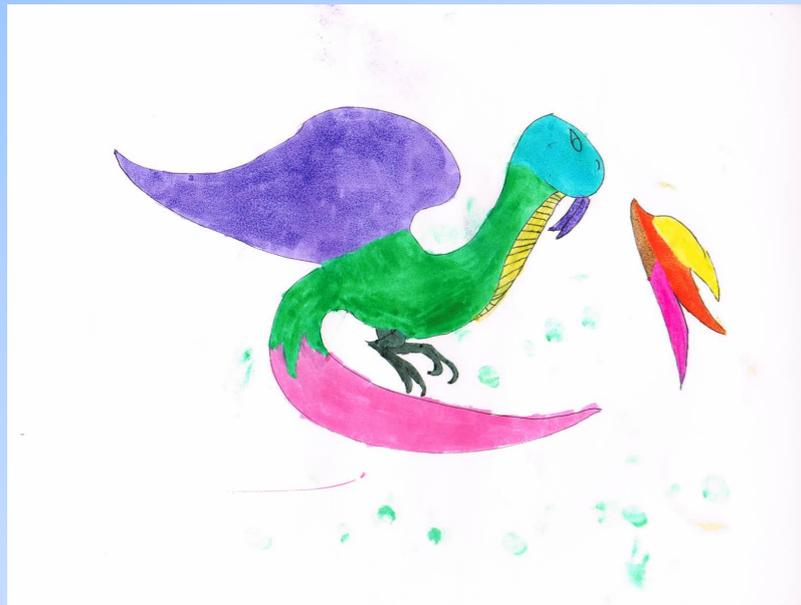
So why do we need another Simulation?

- **Because of output!**
- Other simulations can't drive the **real** aLIGO servos and controls
As aLIGO controls evolve and change, update Simulated Plant just by SVN checkout.
- They don't use the **same software tools** to look at data as aLIGO
- Previous simulations don't work directly with the **CDS**

What is CDS?

- **CDS** stands for "Control and Data Systems"
- Its is a **real-time digital** system
- **Interface** to the analog interferometer (IFO) systems
- **Records** data signals
- Contains many **sub-systems and control loops**

Physical structure of the aLIGO CDS



Analog Realm:
Here be dragons
and IFOs

ADCs



DACs

Digital Realm:
Here be bit noise
and computers

Front End
Computer

DAQ network



Frame
Builder

EPICS

User's
Workstation

NDS and
Raw Frames



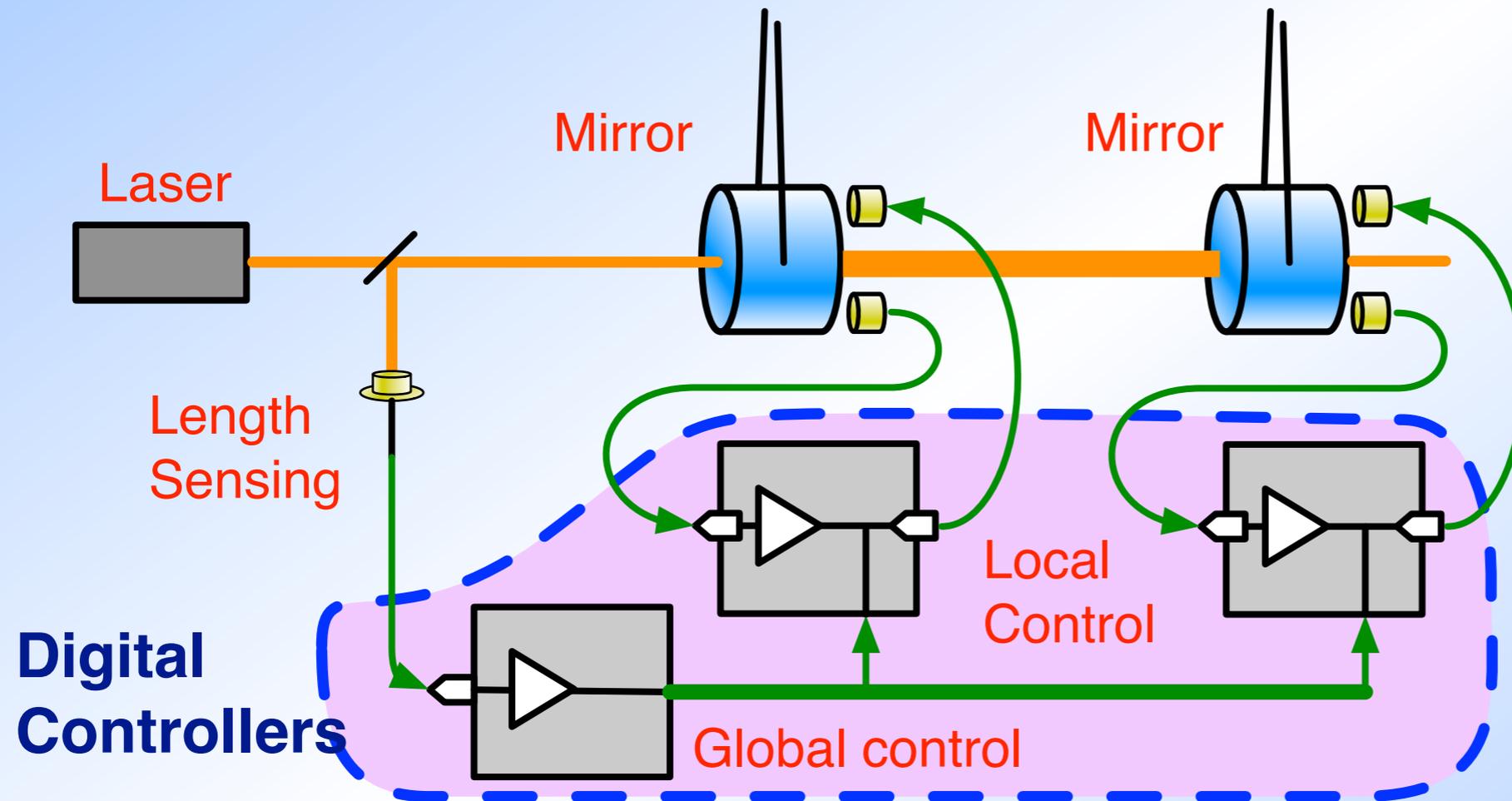
"*Simulated Plant (SP)*"

- **An IFO emulator** for the digital control system
- **A time domain simulator** realized by realtime codes for the digital control system itself
- **Imitating responses** of interferometer components

Basic Idea

Interferometer control:

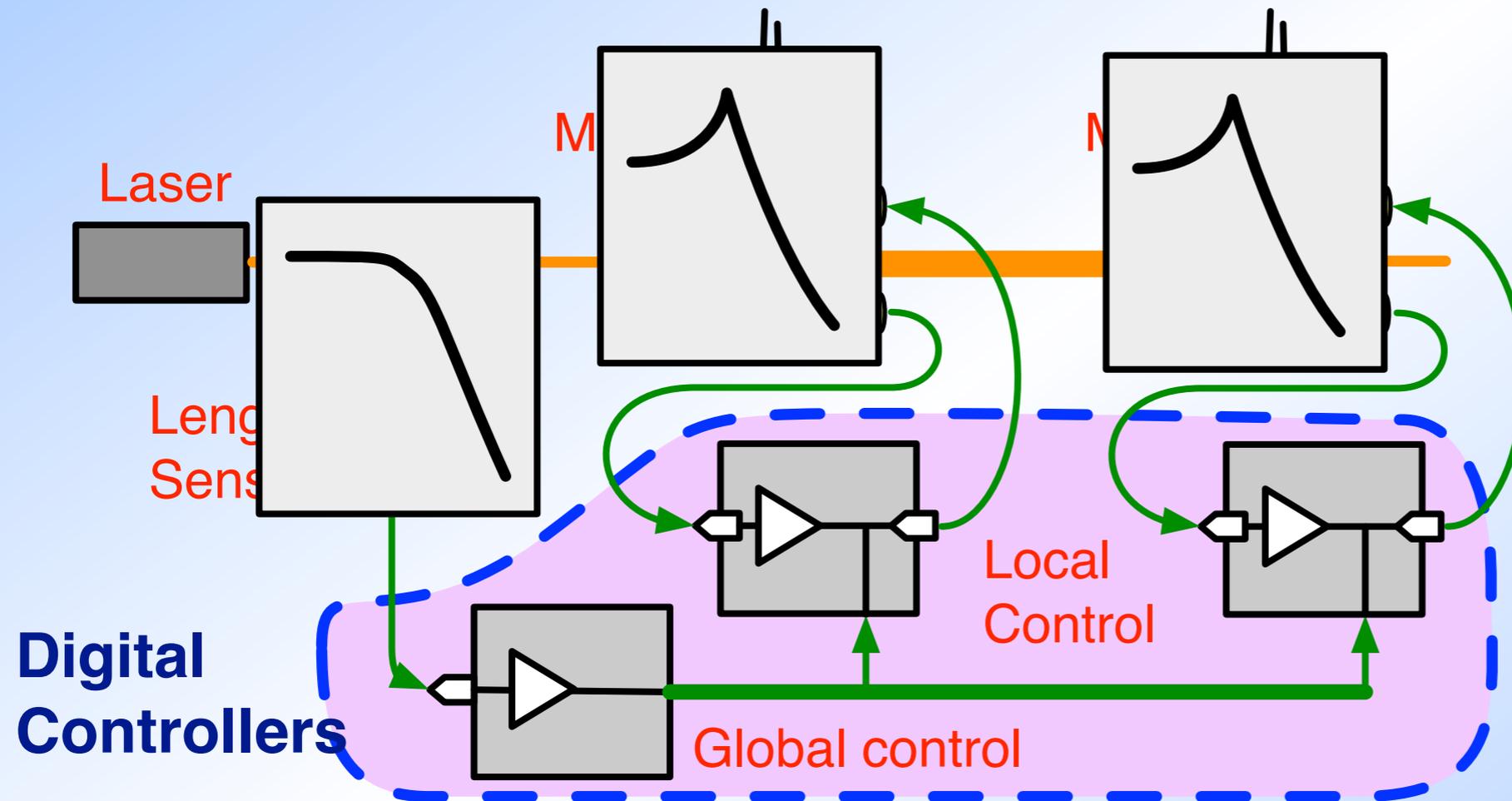
Local control (suspension)
+ Global control (interferometric)



Basic Idea

Interferometer control:

Local control (suspension)
+ Global control (interferometric)

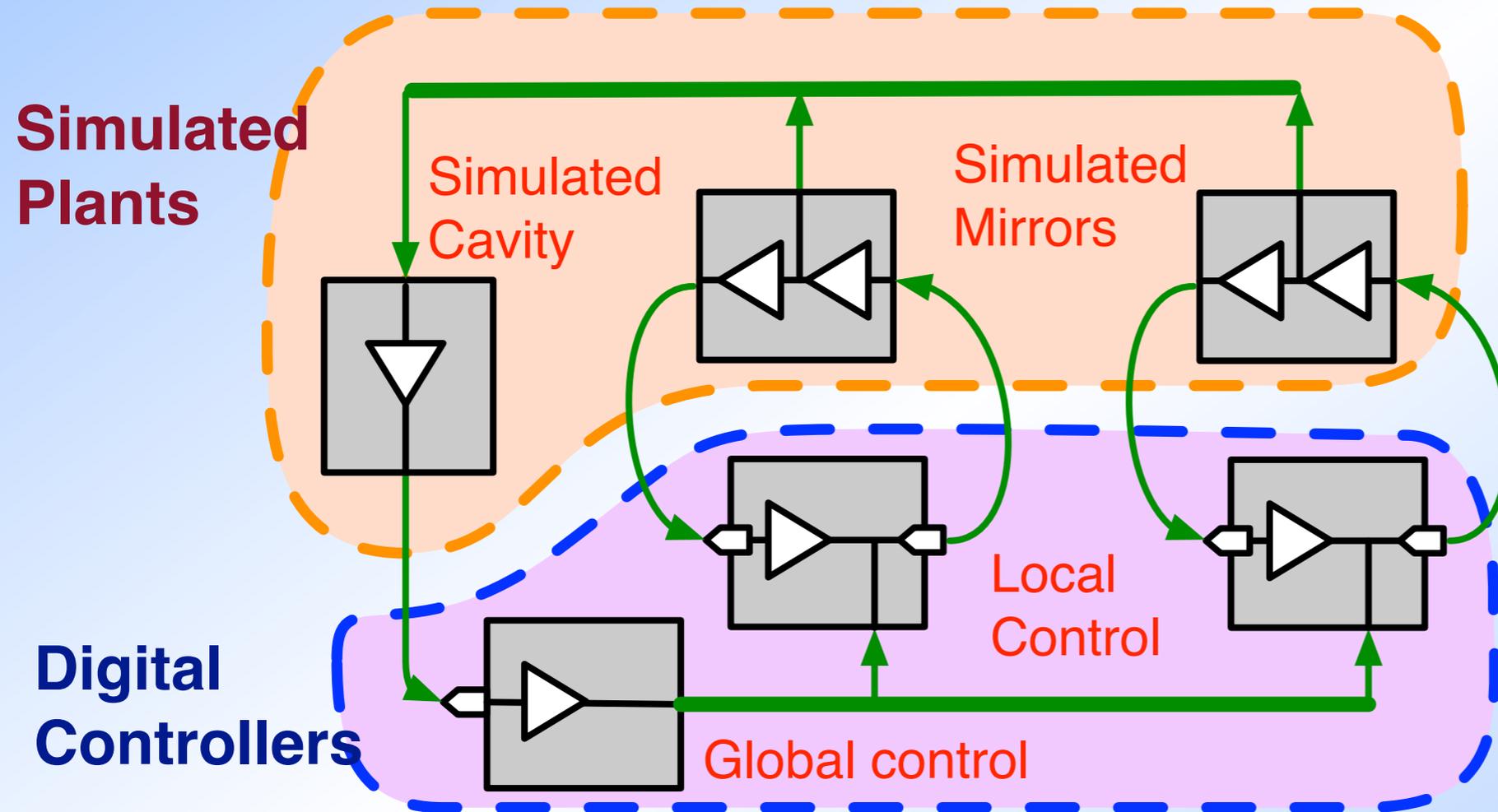


Basic Idea

Replaced hardware responses with digital filters

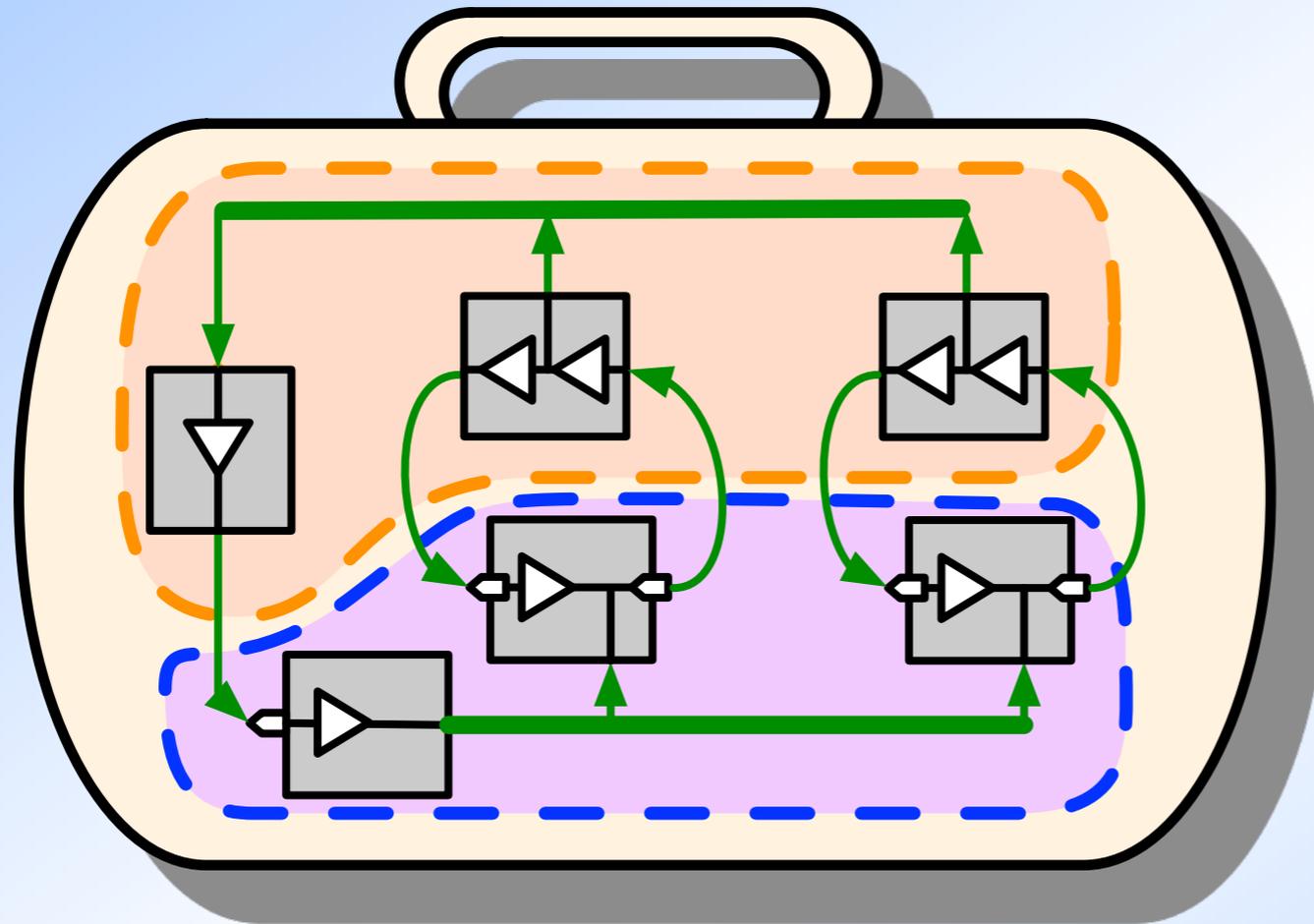
==> **simulated plants**

The servo loops remains stable



Basic Idea

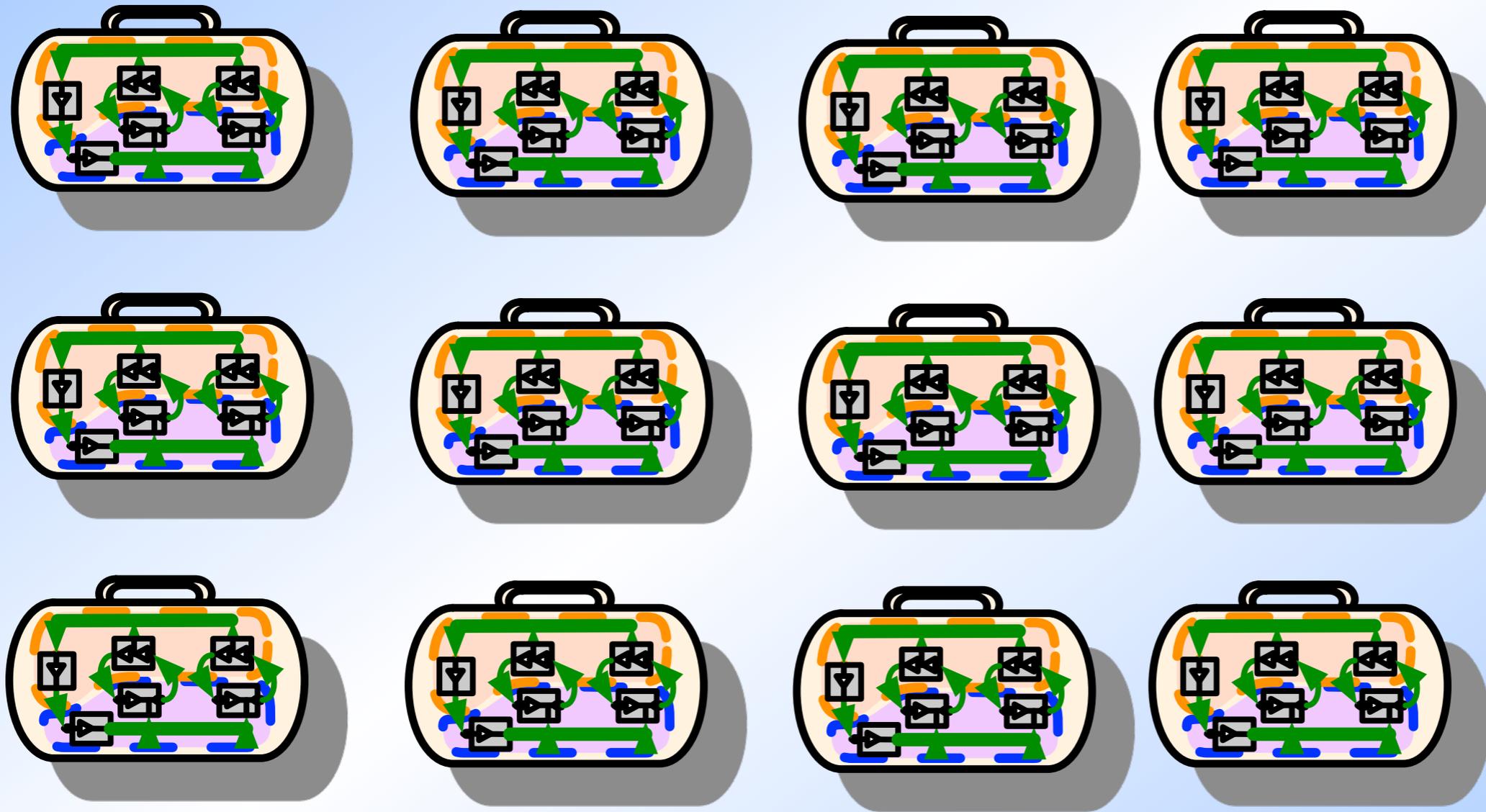
You can put them on a single machine!



Simulated Plant version of the 40m end suspensions

Basic Idea

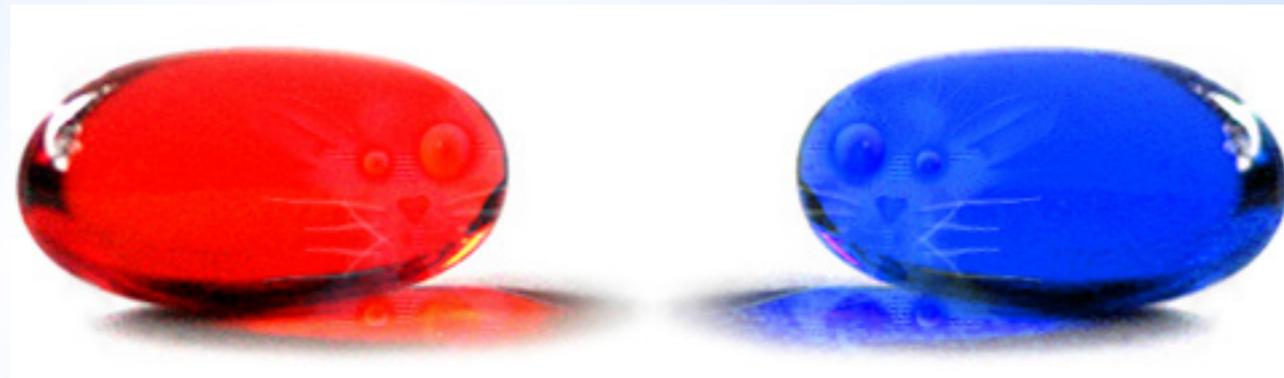
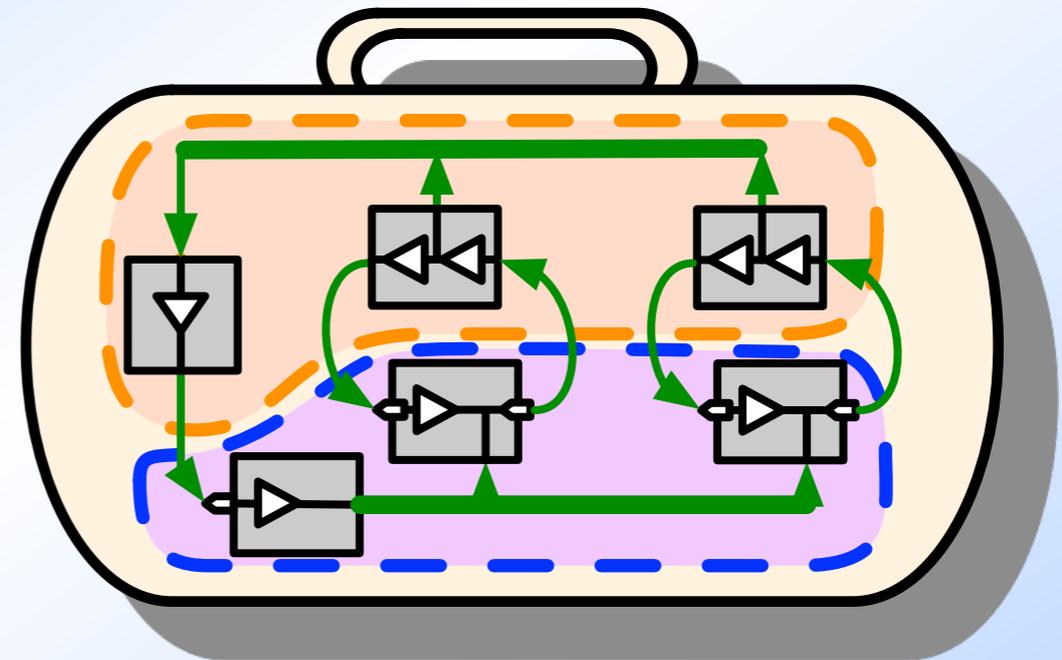
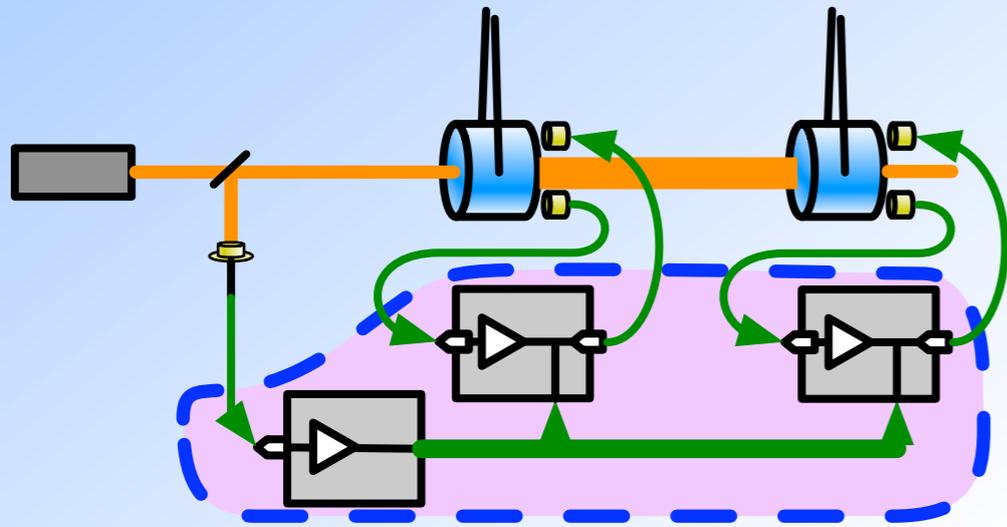
The real CDS is **25 computers** and **100 models** per IFO.
Full simulation could be done with about **12 computers**.



Simulated Plant version of a full LIGO IFO?

Basic Idea

Or you could run the simulation **directly** on those 25 IFO computers. You can then switch between **real** and **simulated** inputs/outputs.



Benefit: Parallel Work

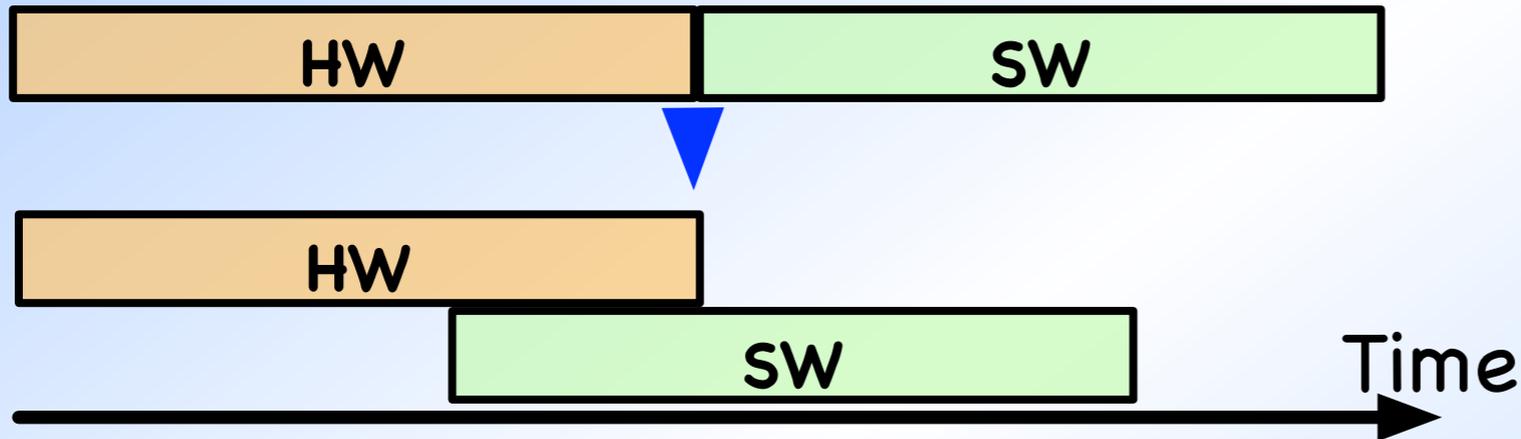
- Commissioning time is valuable

Make the installation and commissioning faster!

- Start building the control SW without actual HWs

While doing single arm test, work on dual recycled IFO code

Realtime codes / Scripts (auto locker / initial alignment)



- Separate controller (computer) problems and SW development from the real HW issues



Even a simple model is sufficient!

Benefit 2: Help understand the noise

- Eventually we face with noise hunting

Make it more convenient

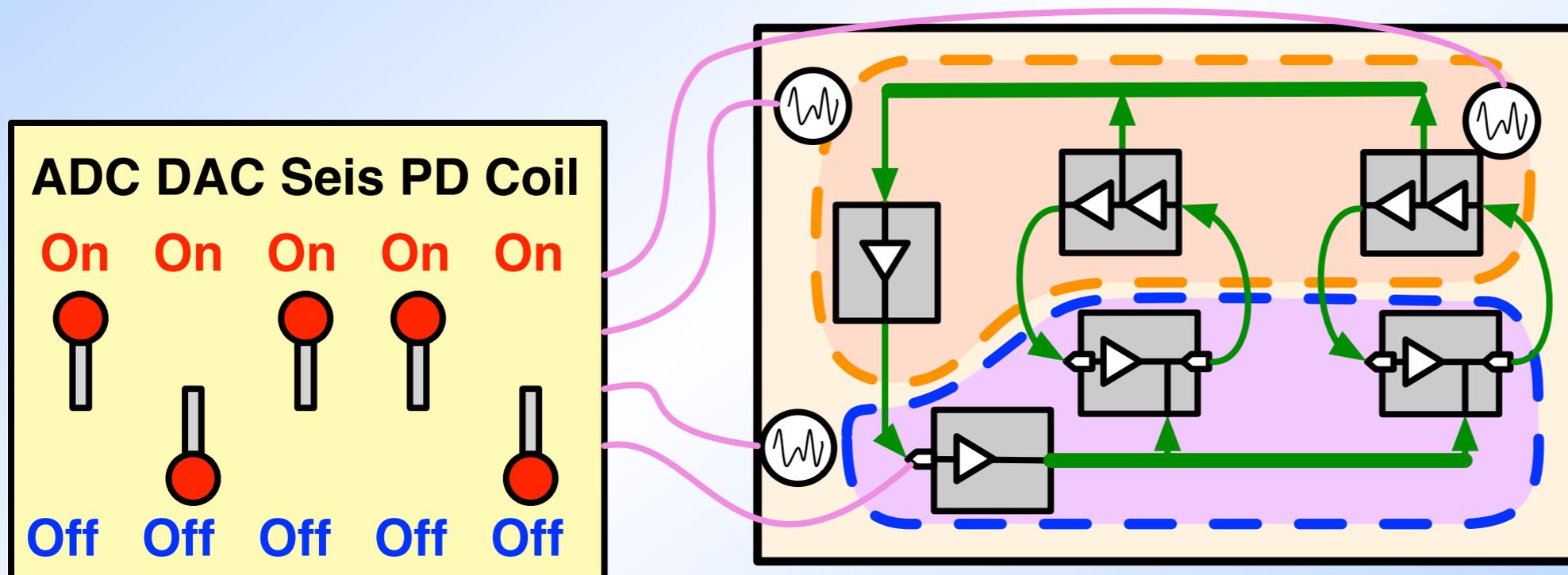
- Build diagnosis SWs offline

Calibration / Measurement templates / Auto noise budget

- Anticipate the noise spectra / saturations

Noise emulation: ADC / DAC / Electronics / Seismic / Laser

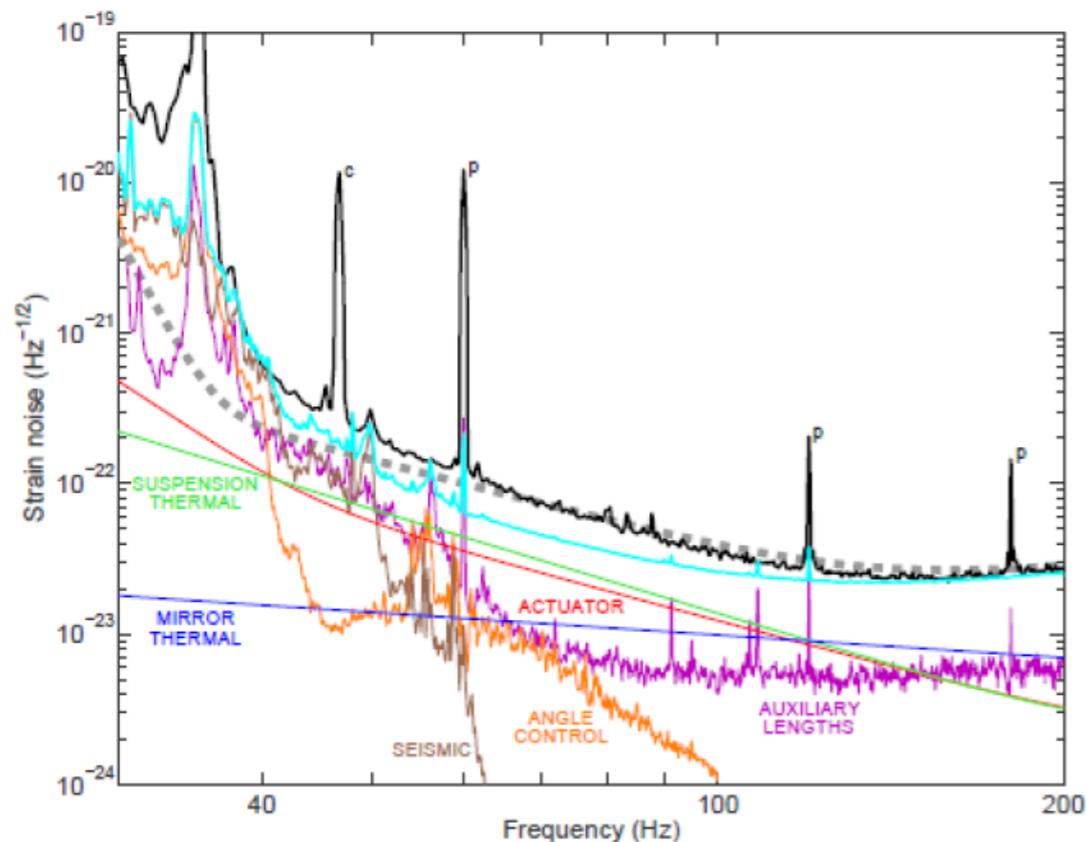
Timing errors / Thermal / Shot / Radiation Pressure



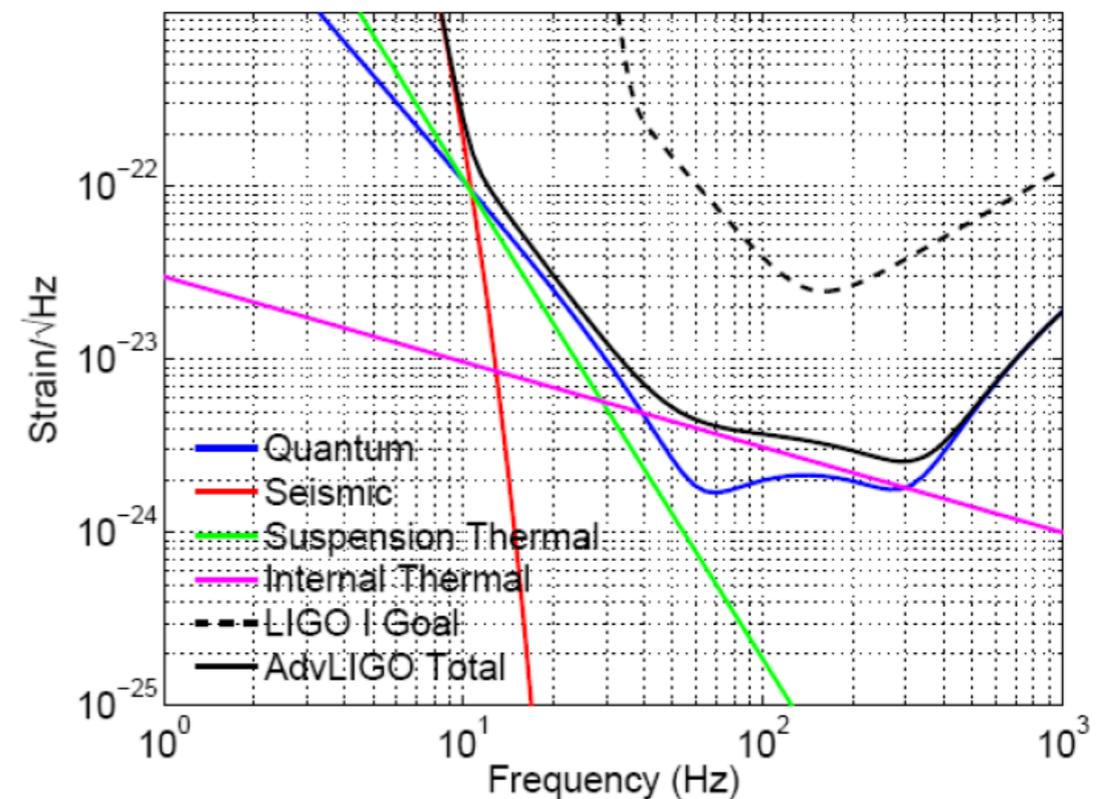
Benefit 2: Help understand the noise

- Initial LIGO noise budgets are extremely complex
 - Predicted aLIGO noise: a few lines with GWINC**
- Simulated Plant can include:
 - ADC saturations, electronics noise, control loops, seismic, thermal, shot, radiation pressure, timing errors**
- Can produce a time series or spectra directly with DTT

Initial LIGO Noise Budget



aLIGO Noise Budget



Benefit 3: Data Analysis

- Does SP only help the commissioning guys?

No! It will be useful to the data analysis people too!

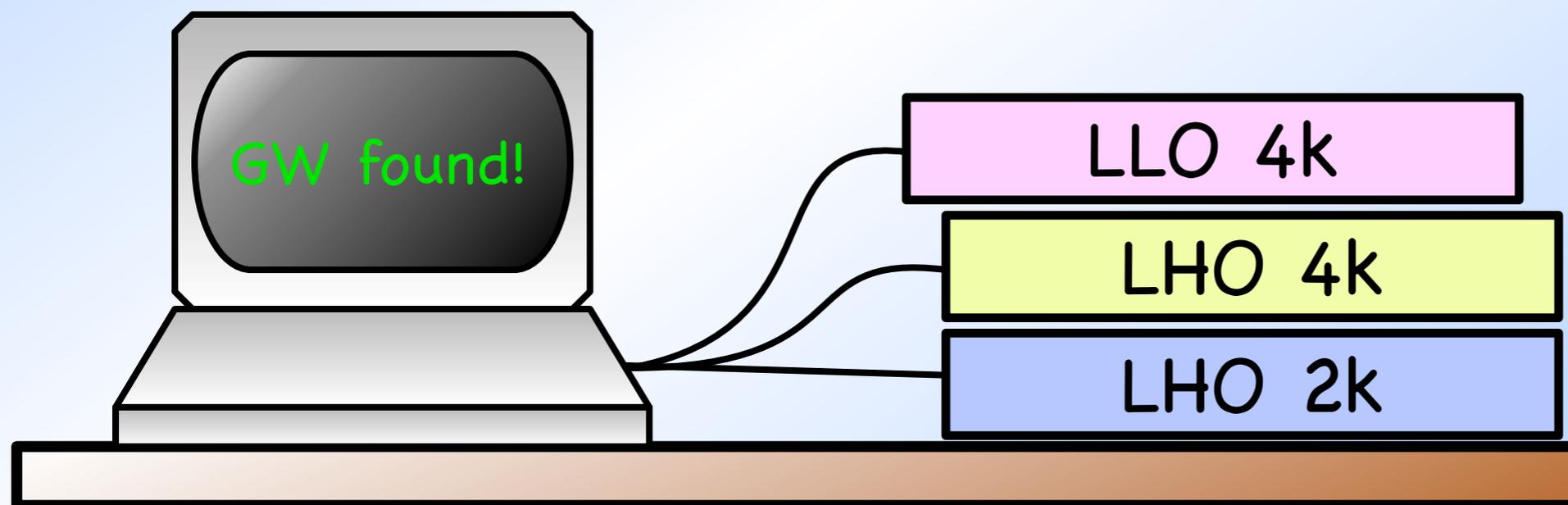
- Imagine a dedicated simulation cluster used for testing:

HW Injection system / DA pipeline / Online monitor

- From Front End to Frame builder to Computering Cluster

All the code will be identical to aLIGO!

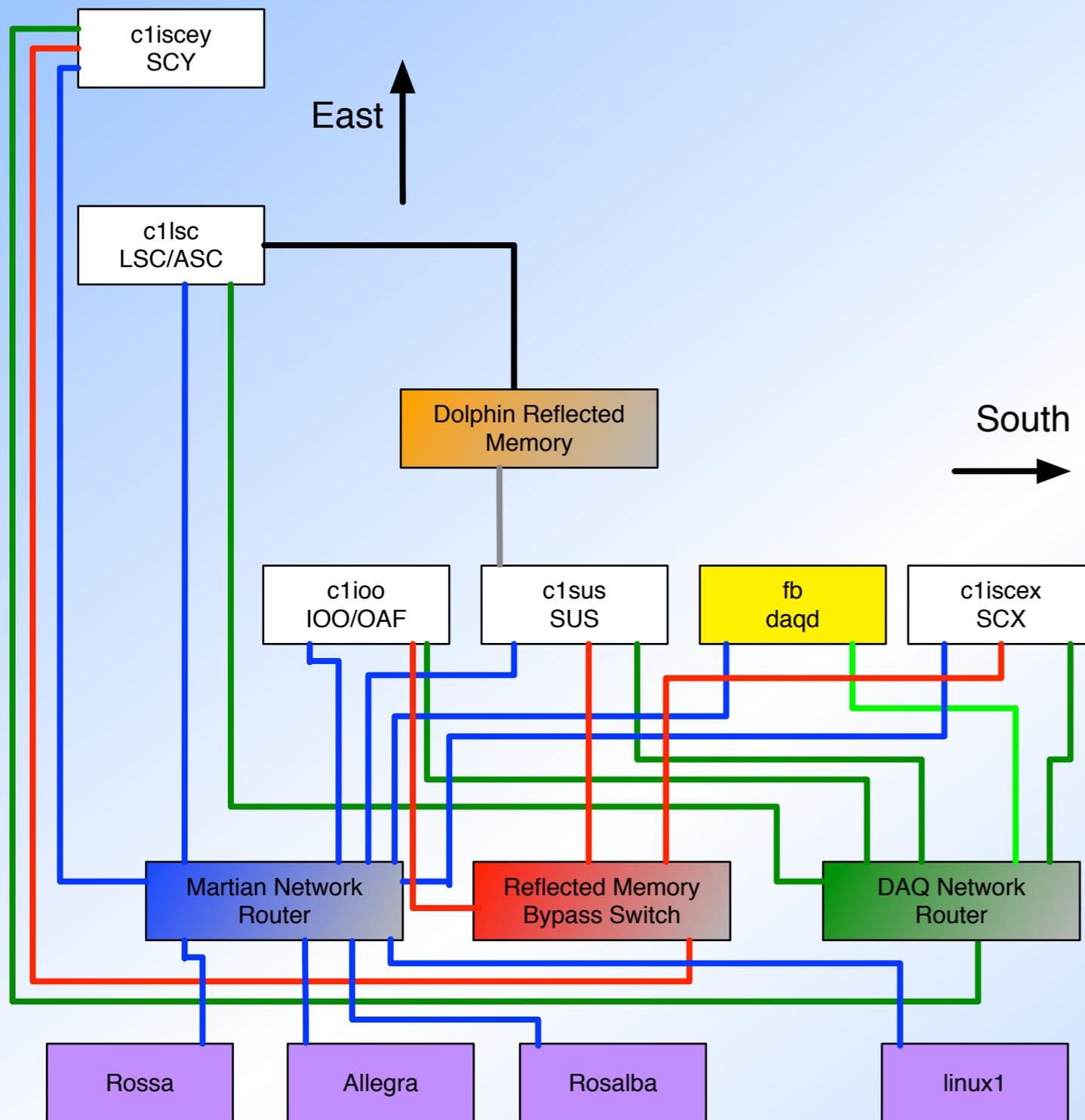
IMAGINE your desk...



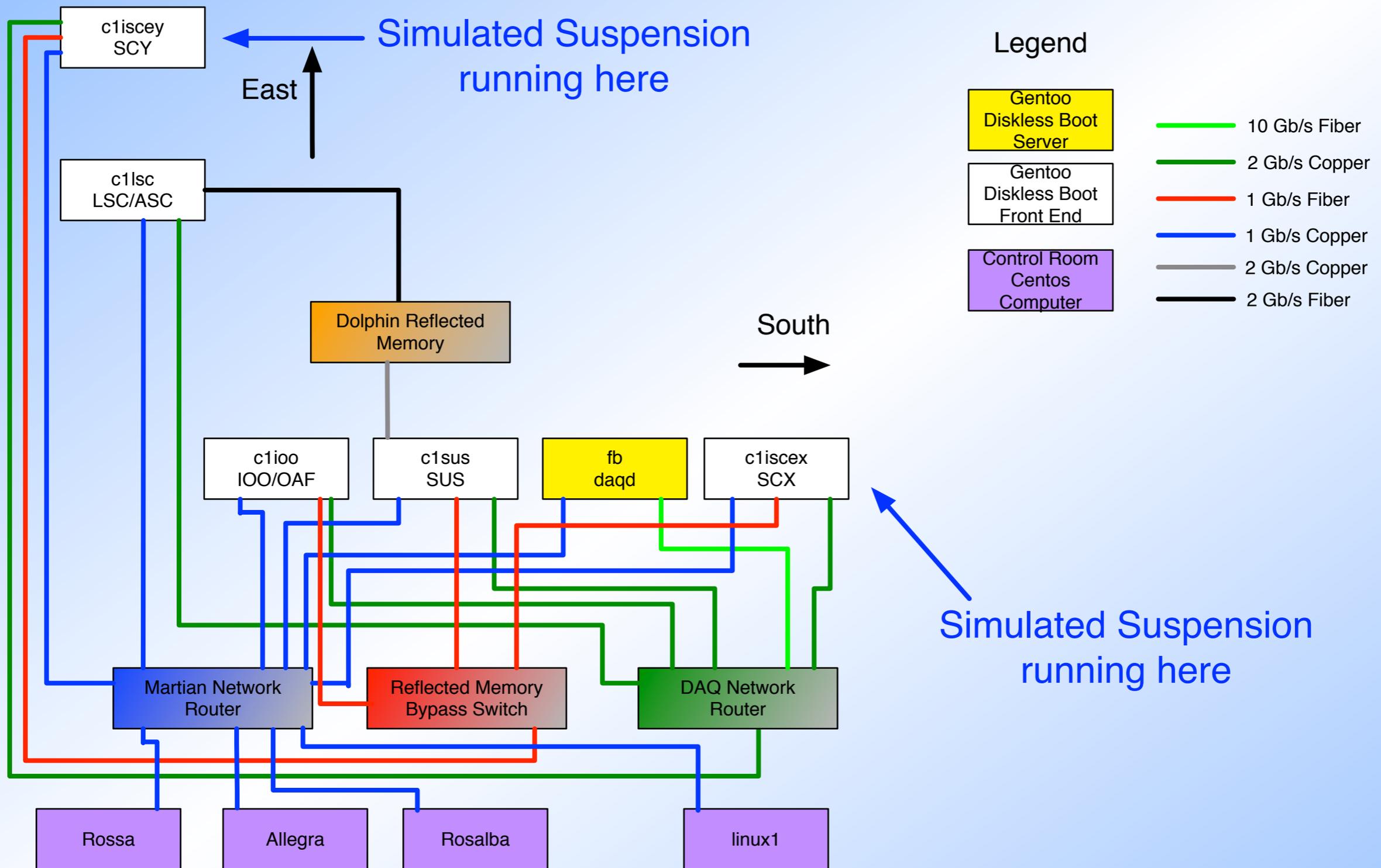
CDS at the 40m

- **40m prototype interferometer** upgraded to the aLIGO CDS
CDS still evolving, upgrading
- 5 front end computers running:
 - 16 control models**
 - 2 simulated plant models (End suspensions)**
- **Frame builder** collecting data from all models and recording

CDS computers and network at the 40m



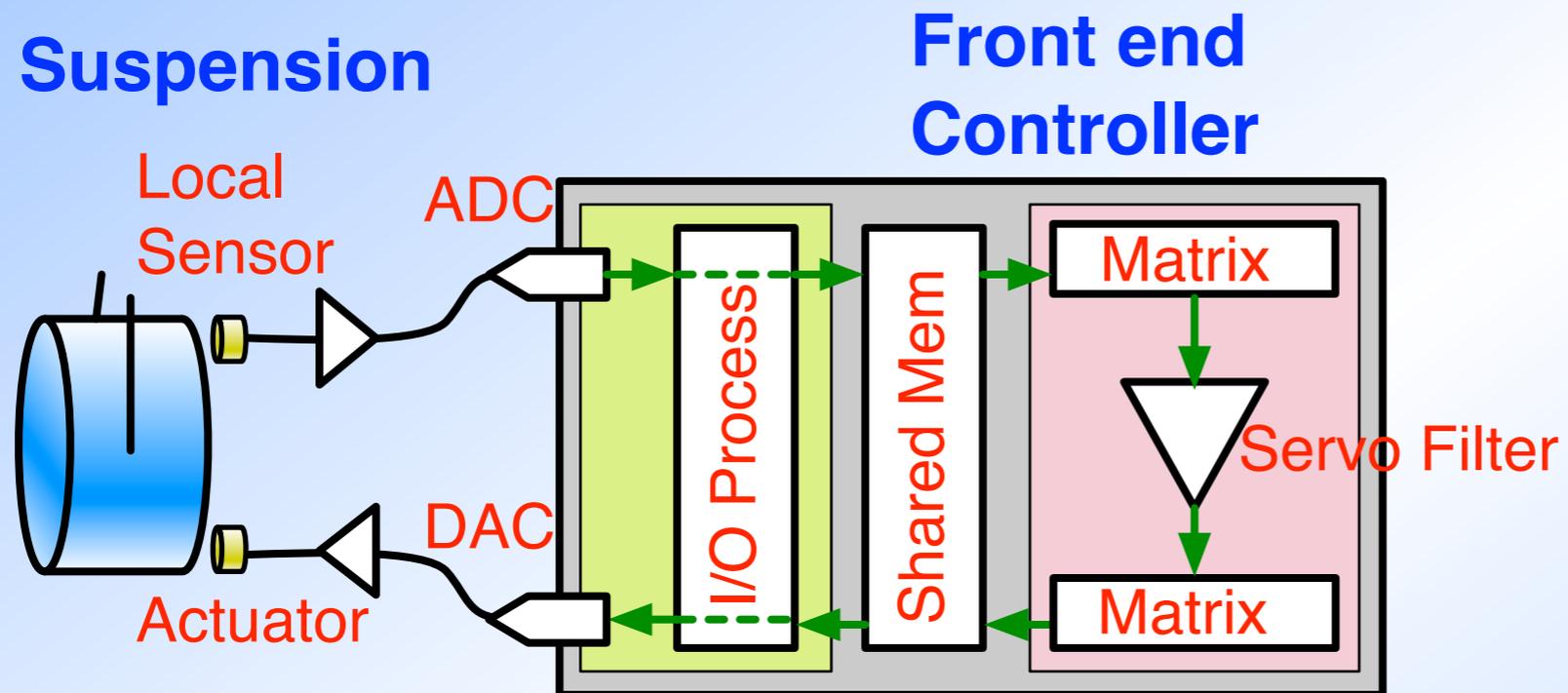
CDS computers and network at the 40m



Simulated Plant: Single Machine Case

- How do we do it? Here's an example.

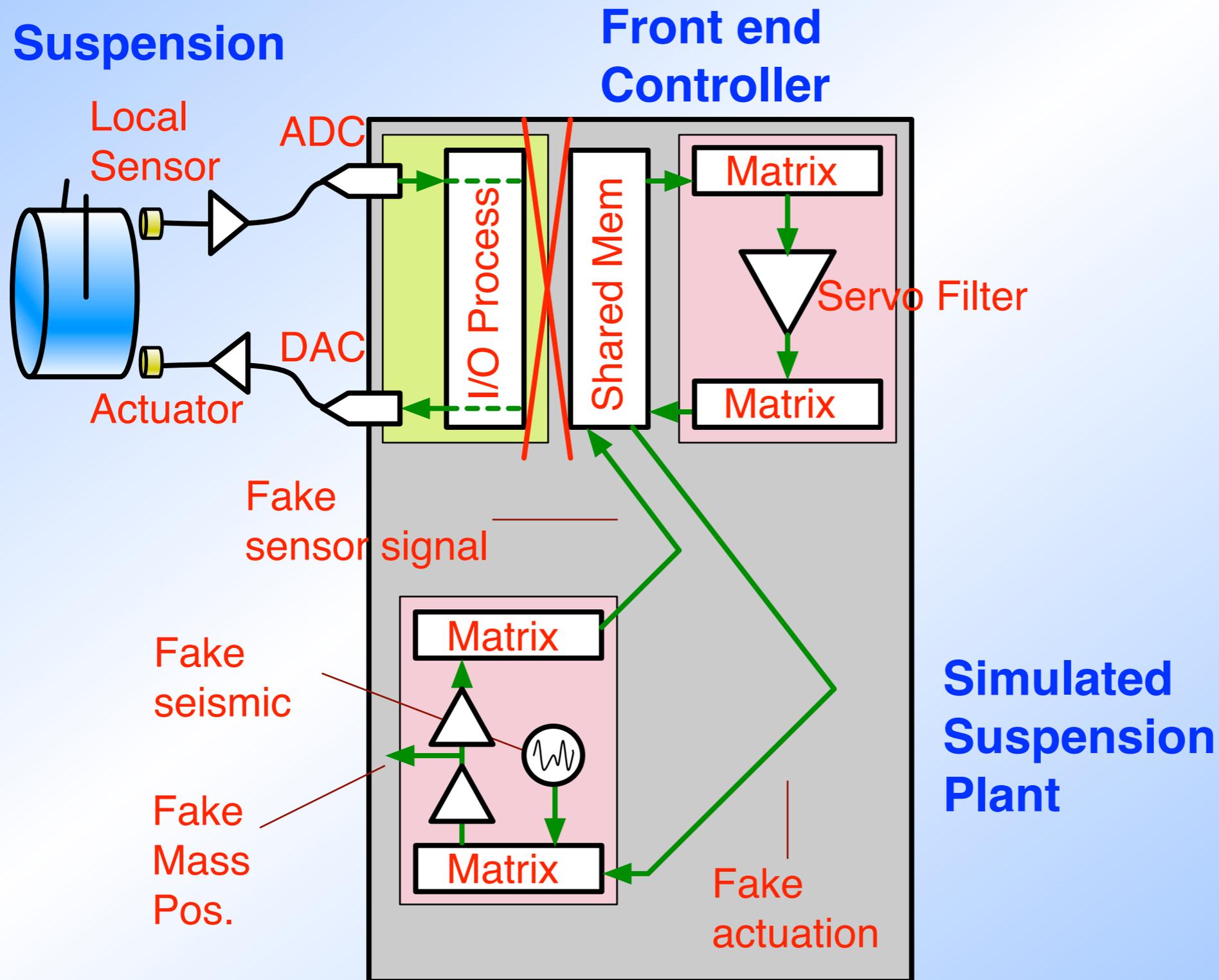
Suspension Controller



Simulated Plant: Single Machine Case

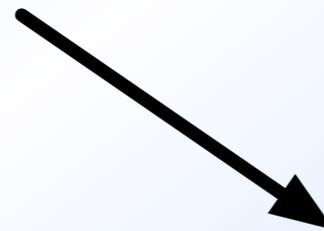
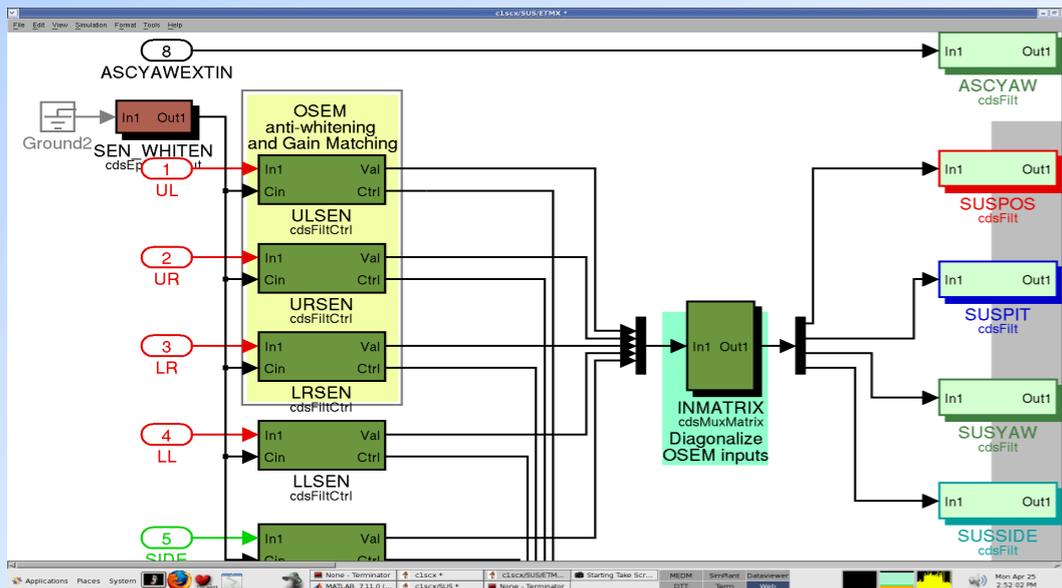
- How do we do it? Here's an example.

Suspension Controller



How to build the models?

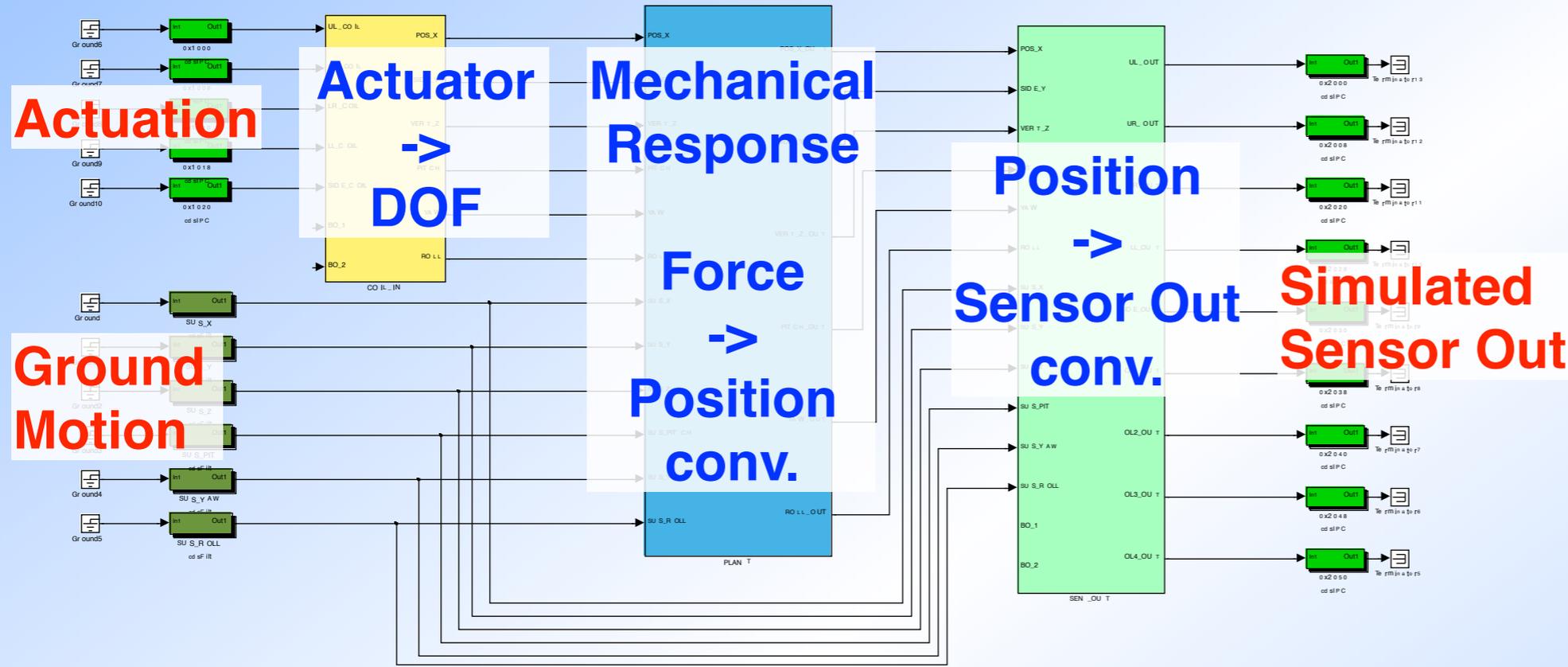
- **RCG: Real-time Code Generator**
 - Use **Simulink** as a GUI to design controls
 - RCG then parses the models to produce **C++** code
 - Code is compiled into **kernel modules** to run on **Front Ends**
 - **Frame builder** is also built to collect and record data



```
ciscx.c - emacs@rosalba
File Edit Options Buffers Tools C Help
// FILTER MODULE with CONTROL: SUS_ETMX_ULSEN
sus_etmx_ulsen = filterModule(dsp_ptr,dspCoeff,SUS_ETMX_ULSEN,choice1,pLocalEpics->scx.SUS_ETMX_SEN_WHITE);
// FILTER MODULE with CONTROL: SUS_ETMX_URSEN
sus_etmx_ursen = filterModule(dsp_ptr,dspCoeff,SUS_ETMX_URSEN,choice3,pLocalEpics->scx.SUS_ETMX_SEN_WHITE);
// FILTER MODULE with CONTROL: SUS_ETMX_LRSEN
sus_etmx_lrse = filterModule(dsp_ptr,dspCoeff,SUS_ETMX_LRSEN,choice4,pLocalEpics->scx.SUS_ETMX_SEN_WHITE);
// FILTER MODULE with CONTROL: SUS_ETMX_LLEN
sus_etmx_llse = filterModule(dsp_ptr,dspCoeff,SUS_ETMX_LLEN,choice2,pLocalEpics->scx.SUS_ETMX_SEN_WHITE);
// FILTER MODULE with CONTROL: SUS_ETMX_SISEN
sus_etmx_sise = filterModule(dsp_ptr,dspCoeff,SUS_ETMX_SISEN,choice,pLocalEpics->scx.SUS_ETMX_SEN_WHITE);
// Switch
sus_etmx_choice2 = (((pLocalEpics->scx.SUS_ETMX_YAW_OFFSET_ON) > 0)? (pLocalEpics->scx.SUS_ETMX_YAW_OFFSET): (sus_etmx_constant7));
// Switch
sus_etmx_choice = (((pLocalEpics->scx.SUS_ETMX_POS_OFFSET_ON) > 0)? (pLocalEpics->scx.SUS_ETMX_POS_OFFSET): (sus_etmx_constant7));
// EpicsOut: SUS_ETMX_LOCKIN1_CLOCK
pLocalEpics->scx.SUS_ETMX_LOCKIN1_CLOCK = sus_etmx_lockin1_osc[0];
// EpicsOut: SUS_ETMX_LOCKIN2_CLOCK
pLocalEpics->scx.SUS_ETMX_LOCKIN2_CLOCK = sus_etmx_lockin2_osc[0];
// MUX
sus_etmx_mux2[0] = sus_etmx_o11;
sus_etmx_mux2[1] = sus_etmx_o12;
sus_etmx_mux2[2] = sus_etmx_o13;
sus_etmx_mux2[3] = sus_etmx_o14;
ciscx.c (C Abbrev)--L603--43
```

Actual Suspension Simulated Plant

Formed by matrices and arrays(or matrices) of filter modules

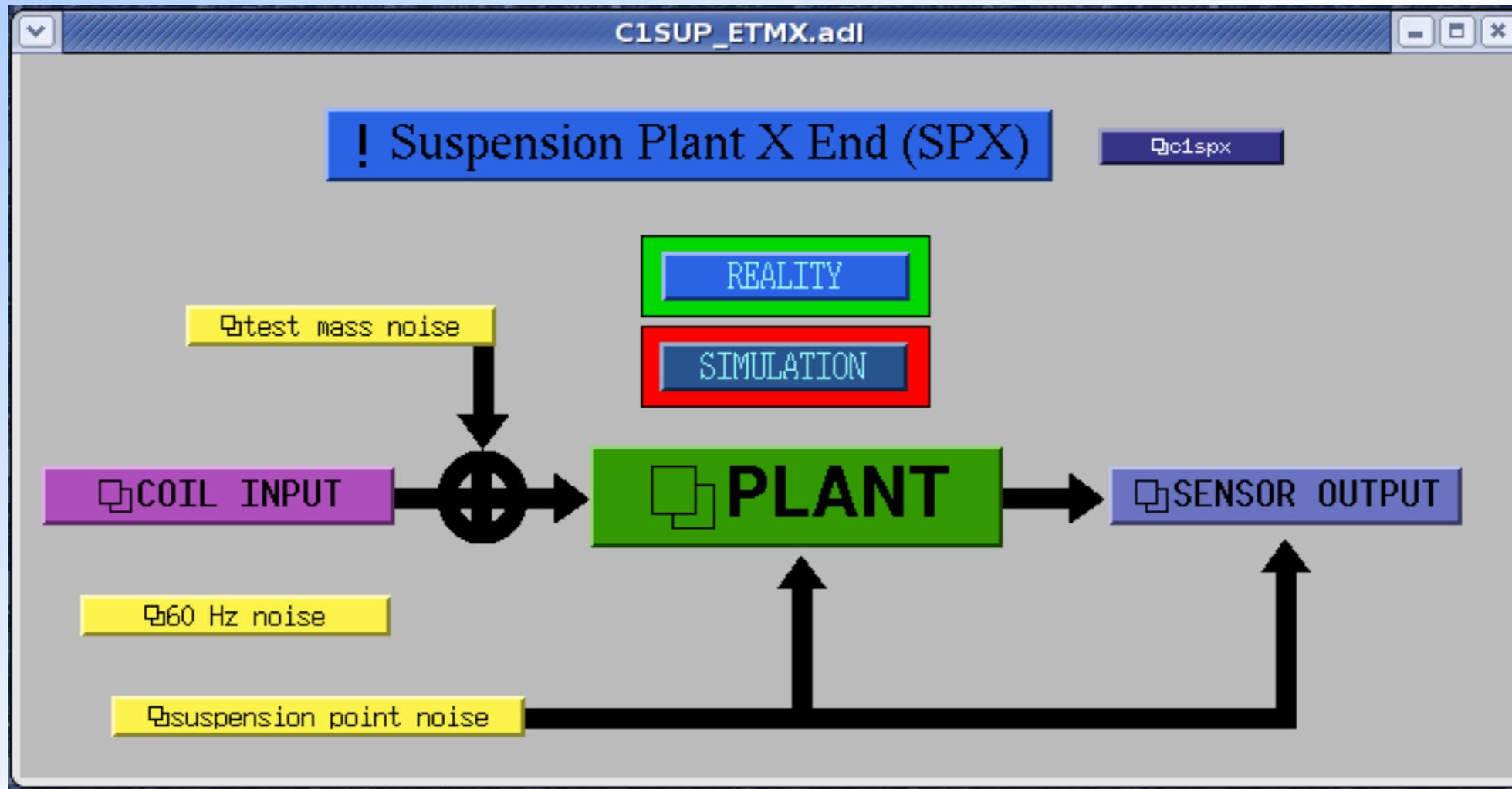


Actual Suspension Simulated Plant

Formed by matrices and arrays(or matrices) of filter modules

Physics:
Mech. responses = transfer functions
==> the responses are realized
by the filters and matrix elements
Once the model is built, it is flexible
i.e. single pendulum, quad pendulum,
etc...

40M Simulated Plant Control Screens



40M Simulated Plant Control Screens

C1SUP_ETMX_CI.adl

! Suspension Plant X End (SPX) - Coil Input

DAC volts

Tue Apr 26 10:19:56

UL UR LR LL SIDE

0.061 0.141 0.781 0.946 0.141 force (newtons)

0.250	0.250	0.250	0.250	0.000	X/POS
0.000	0.000	0.000	0.000	1.000	Y/SIDE
0.000	0.000	0.000	0.000	0.000	Z/VERT
0.006	0.006	-0.006	-0.006	0.000	PIT
0.006	-0.006	-0.006	0.006	0.000	YAW
0.000	0.000	0.000	0.000	0.000	ROLL

! Quick Matrix

test mass force/torque
(newtons/newton-meters)

40M Simulated Plant Control Screens

C1SUP_ETMX_PLANT.adl

! Suspension Plant X End (SPX) - Physical Plant

test mass coil force/torque
(newtons/newton-meters)

suspension point position/angle
(meters/radians)

X/POS	Y/SIDE	Z/VERT	PIT	YAW	ROLL	X/POS	Y/SIDE	Z/VERT	PIT	YAW	ROLL
1.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000
☒1_1	☒1_2	☒1_3	☒1_4	☒1_5	☒1_6	☒1_7	☒1_8	☒1_9	☒1_10	☒1_11	☒1_12
0.000	1.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000
☒2_1	☒2_2	☒2_3	☒2_4	☒2_5	☒2_6	☒2_7	☒2_8	☒2_9	☒2_10	☒2_11	☒2_12
0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000
☒3_1	☒3_2	☒3_3	☒3_4	☒3_5	☒3_6	☒3_7	☒3_8	☒3_9	☒3_10	☒3_11	☒3_12
0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
☒4_1	☒4_2	☒4_3	☒4_4	☒4_5	☒4_6	☒4_7	☒4_8	☒4_9	☒4_10	☒4_11	☒4_12
0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
☒5_1	☒5_2	☒5_3	☒5_4	☒5_5	☒5_6	☒5_7	☒5_8	☒5_9	☒5_10	☒5_11	☒5_12
0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000
☒6_1	☒6_2	☒6_3	☒6_4	☒6_5	☒6_6	☒6_7	☒6_8	☒6_9	☒6_10	☒6_11	☒6_12

▶ X/POS

▶ Y/SIDE

▶ Z/VERT

▶ PIT

▶ YAW

▶ ROLL

test mass position/angle
(meters/radians)

! Quick Matrix

40M Simulated Plant Control Screens

C1SUP_ETMX_SO.adl Tue Apr 26 10:19:43

! Suspension Plant X End (SPX) - Sensor Outputs

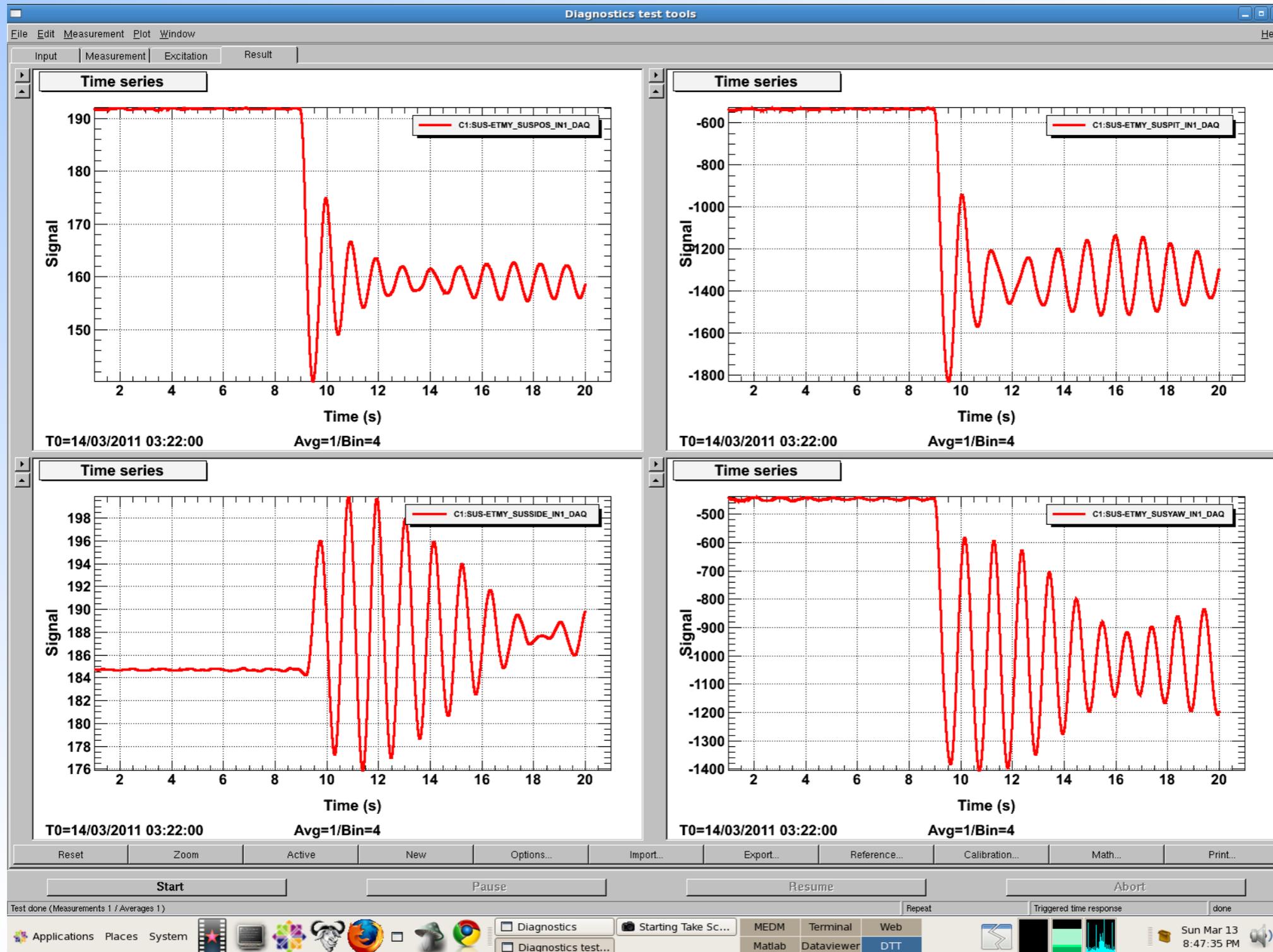
test mass position/angle (meters/radians) suspension point position/angle (meters/radians)

POS	SIDE	VERT	PIT	YAW	ROLL	POS	SIDE	VERT	PIT	YAW	ROLL	
1.000	0.000	0.000	1.000	1.000	0.000	-1.00	0.000	0.000	-1.00	-1.00	0.000	▶ <input type="checkbox"/> UL
1.000	0.000	0.000	1.000	-1.00	0.000	-1.00	0.000	0.000	-1.00	1.000	0.000	▶ <input type="checkbox"/> UR
1.000	0.000	0.000	-1.00	-1.00	0.000	-1.00	0.000	0.000	1.000	1.000	0.000	▶ <input type="checkbox"/> LR
1.000	0.000	0.000	-1.00	1.000	0.000	-1.00	0.000	0.000	1.000	-1.00	0.000	▶ <input type="checkbox"/> LL
0.000	1.000	0.000	0.000	0.000	0.000	0.000	-1.00	0.000	0.000	0.000	0.000	▶ <input type="checkbox"/> SIDE
0.000	0.000	0.000	-1.00	-1.00	0.000	0.000	0.000	0.000	0.000	0.000	0.000	▶ <input type="checkbox"/> OL1
0.000	0.000	0.000	1.000	-1.00	0.000	0.000	0.000	0.000	0.000	0.000	0.000	▶ <input type="checkbox"/> OL2
0.000	0.000	0.000	1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	▶ <input type="checkbox"/> OL3
0.000	0.000	0.000	-1.00	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	▶ <input type="checkbox"/> OL4

! Quick Matrix sensor outputs (ADC)

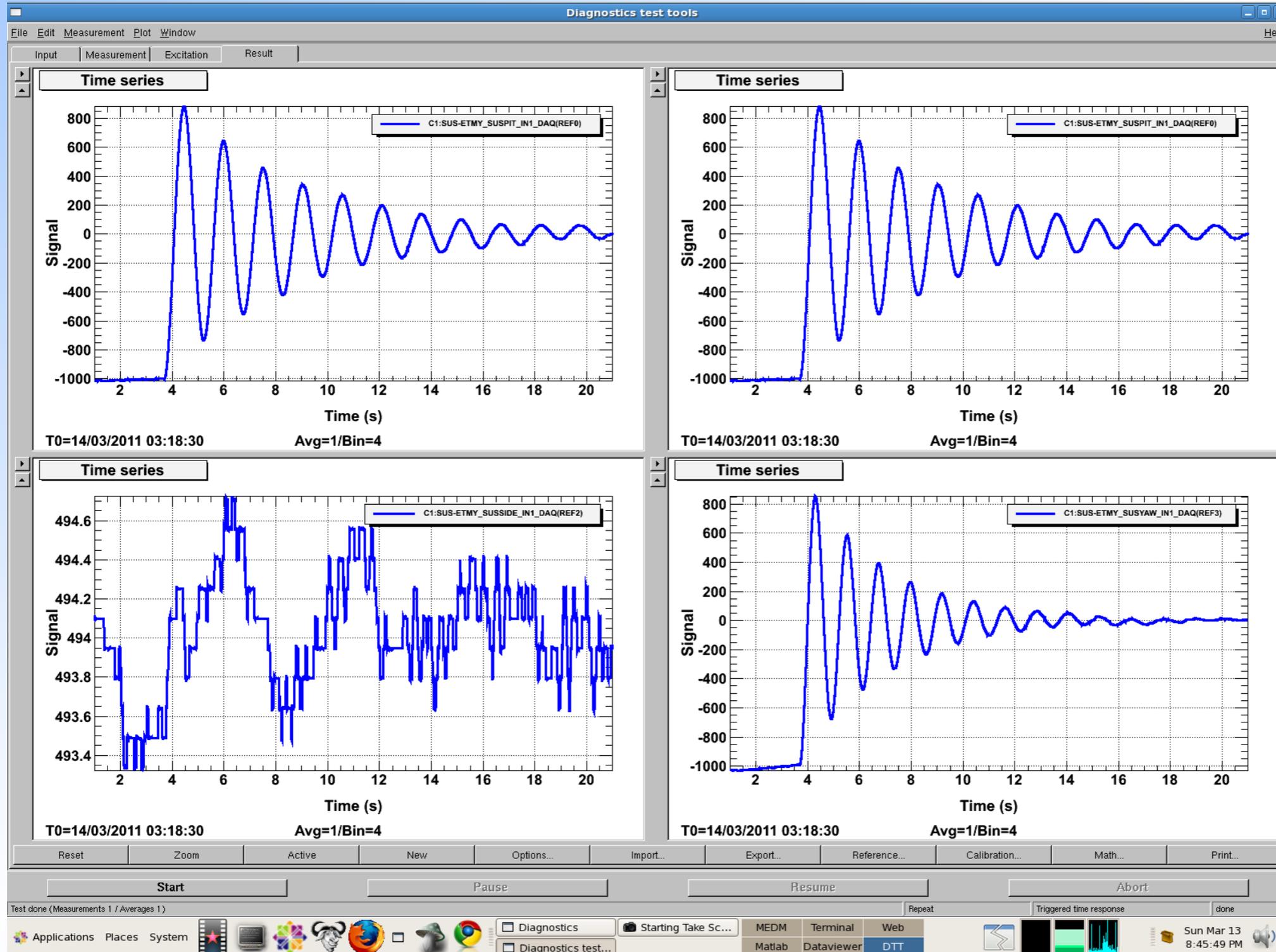
40M Simulated Plant Real Damping

- Data taken at the 40m with the normal commissioning tools

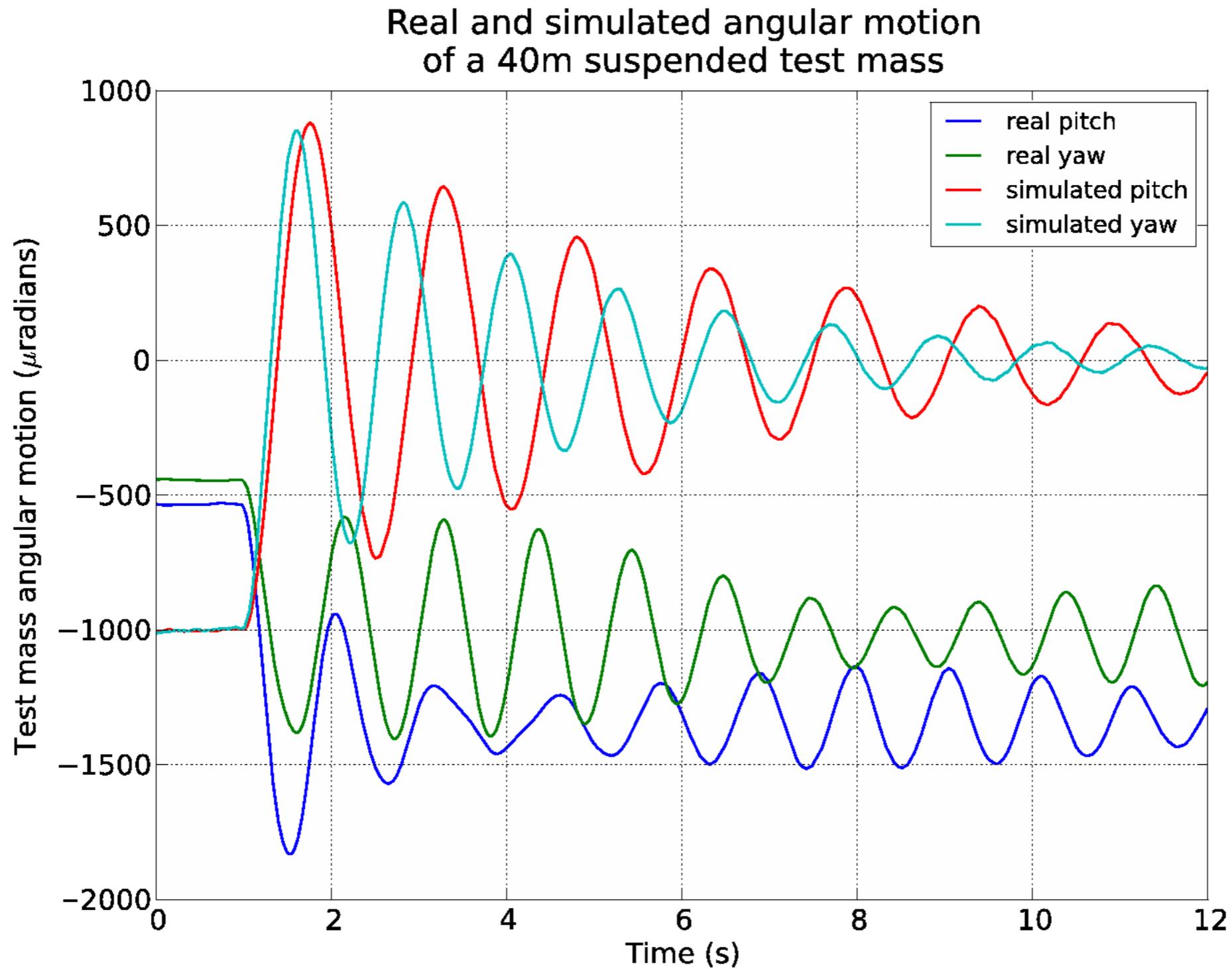


40M Simulated Plant Simulated Damping

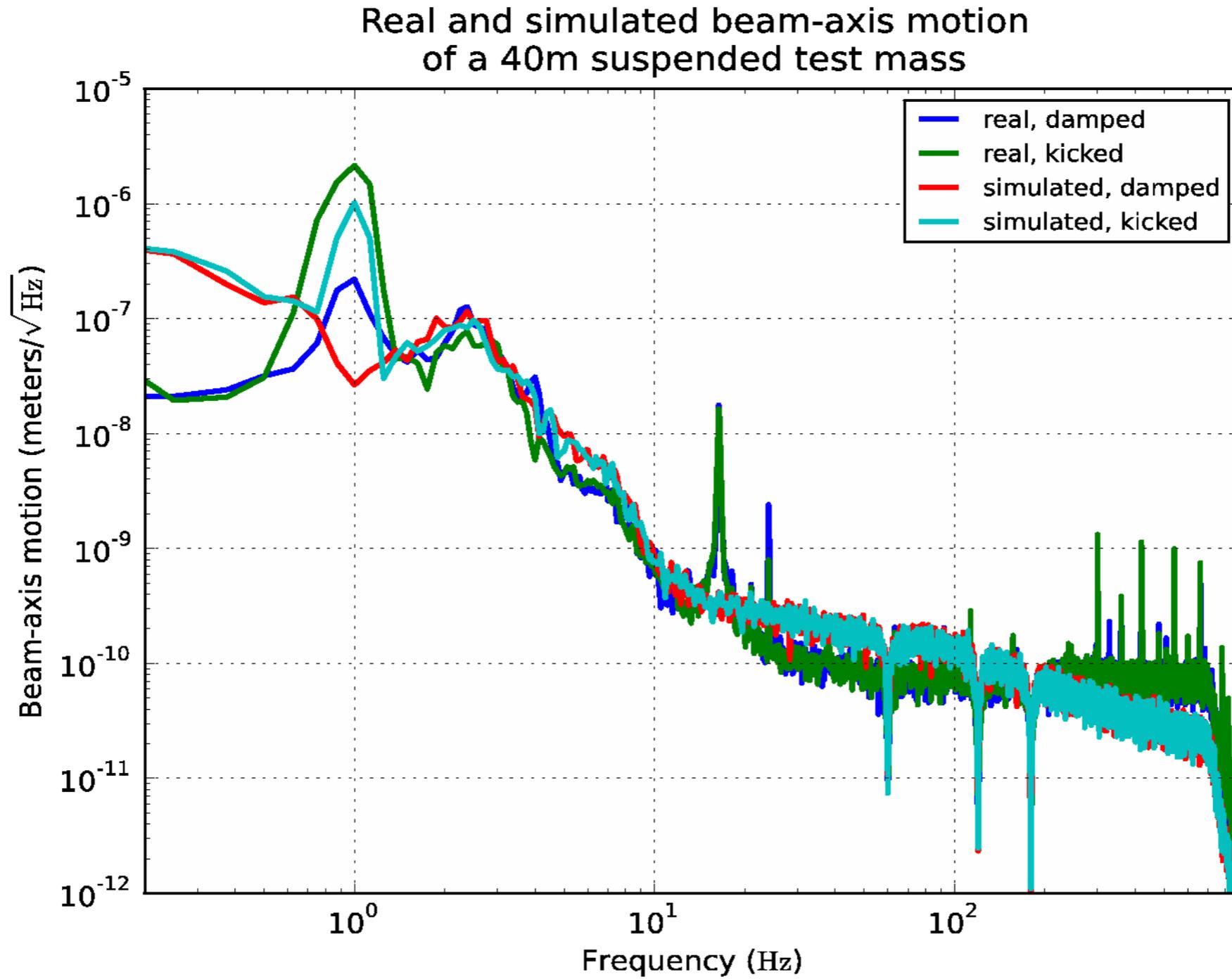
- Simulated data taken a few moments later with the same tools



40M Simulated Plant Damping Comparison



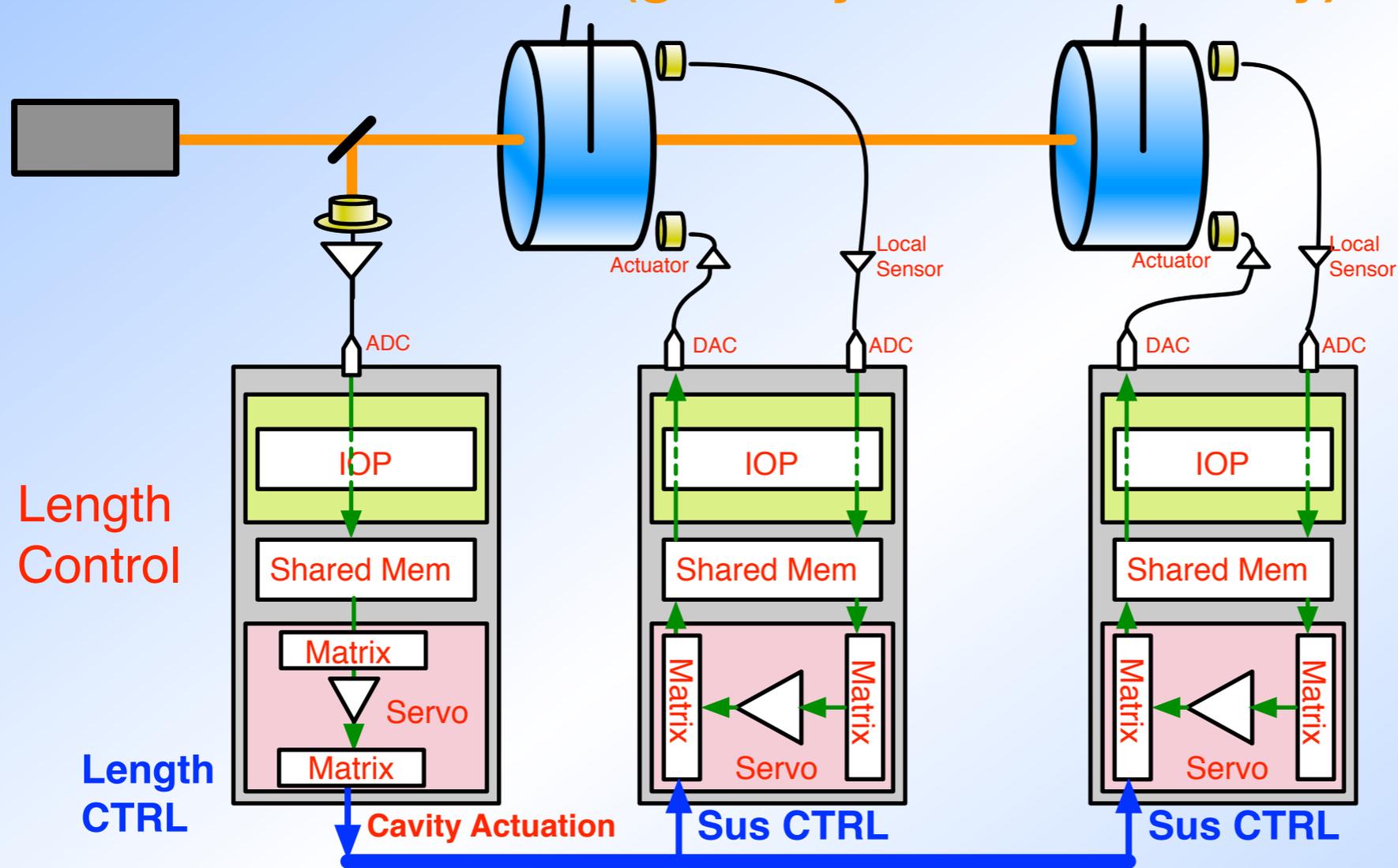
40M Simulated Plant Suspension Noise



Simulated Plant: multiple machine case

Length Controller ~ Multiple machine case

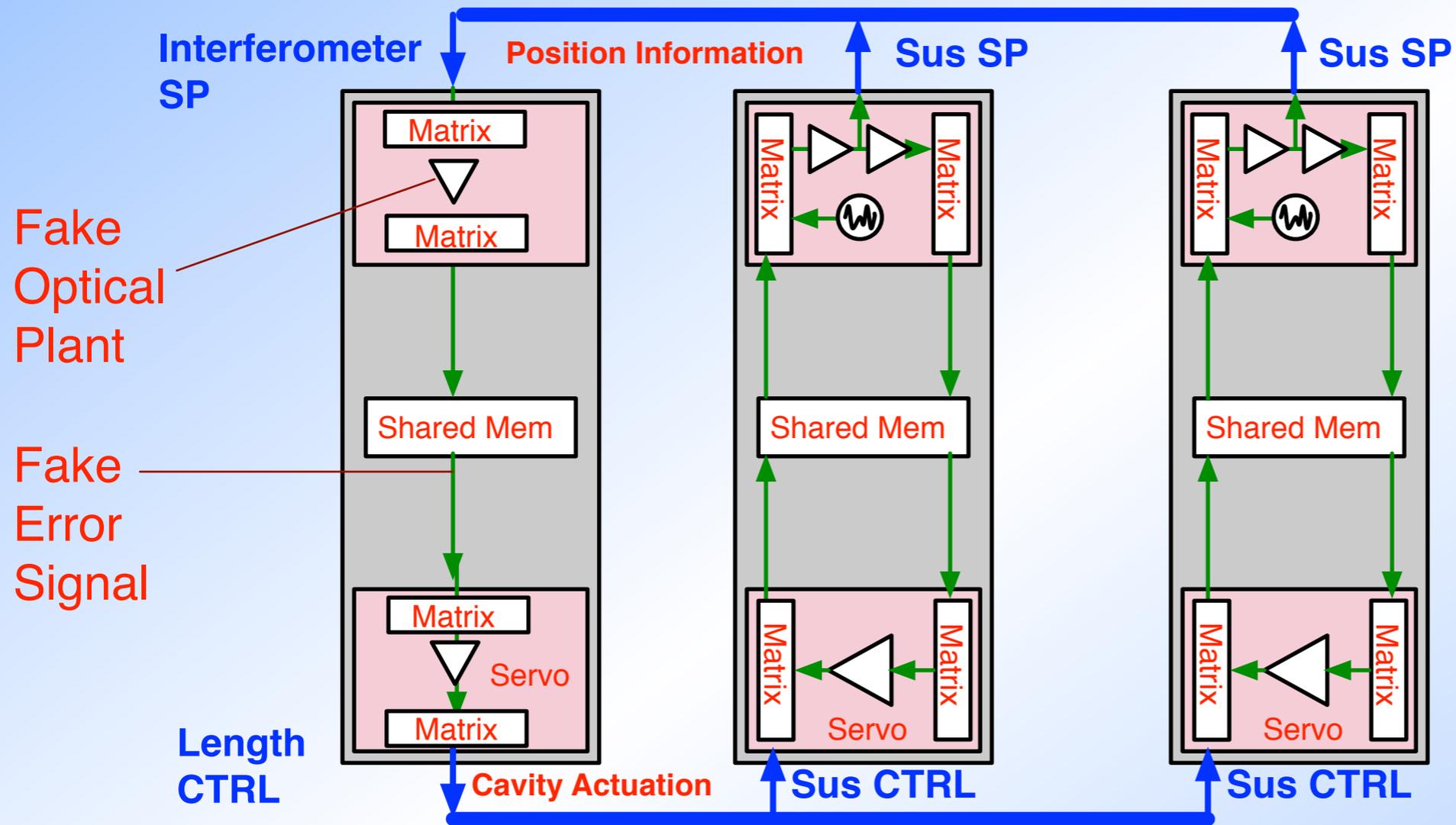
RFM/PCIe communication (globally-shared memory)



Simulated Plant: multiple machine case

Length Controller ~ Multiple machine case

RFM/PCIe communication (globally-shared memory)



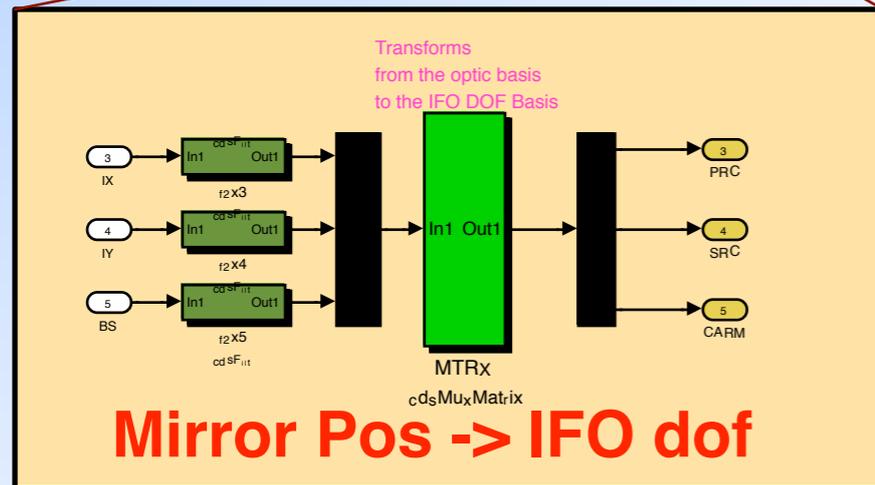
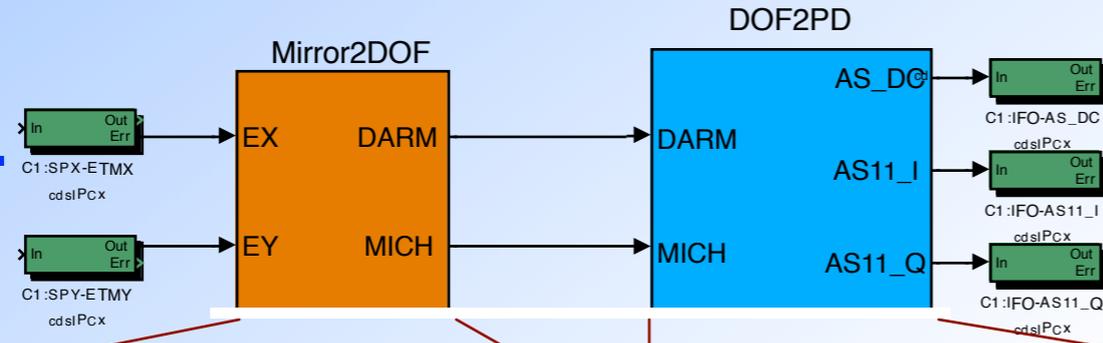
IFO Simulated Plant

Formed by matrices and arrays(or matrices) of filter modules

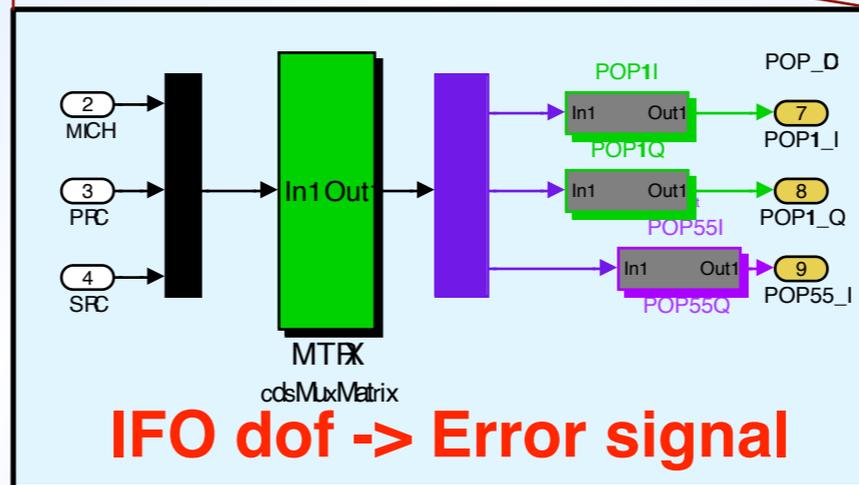
40m LSC Plant

Mirror Pos.
info

Simulated
Error signal



Mirror Pos -> IFO dof



IFO dof -> Error signal

IFO Simulated Plant

Formed by matrices and arrays(or matrices) of filter modules

40m LSC Plant

Mirror Pos.
info

Physics:

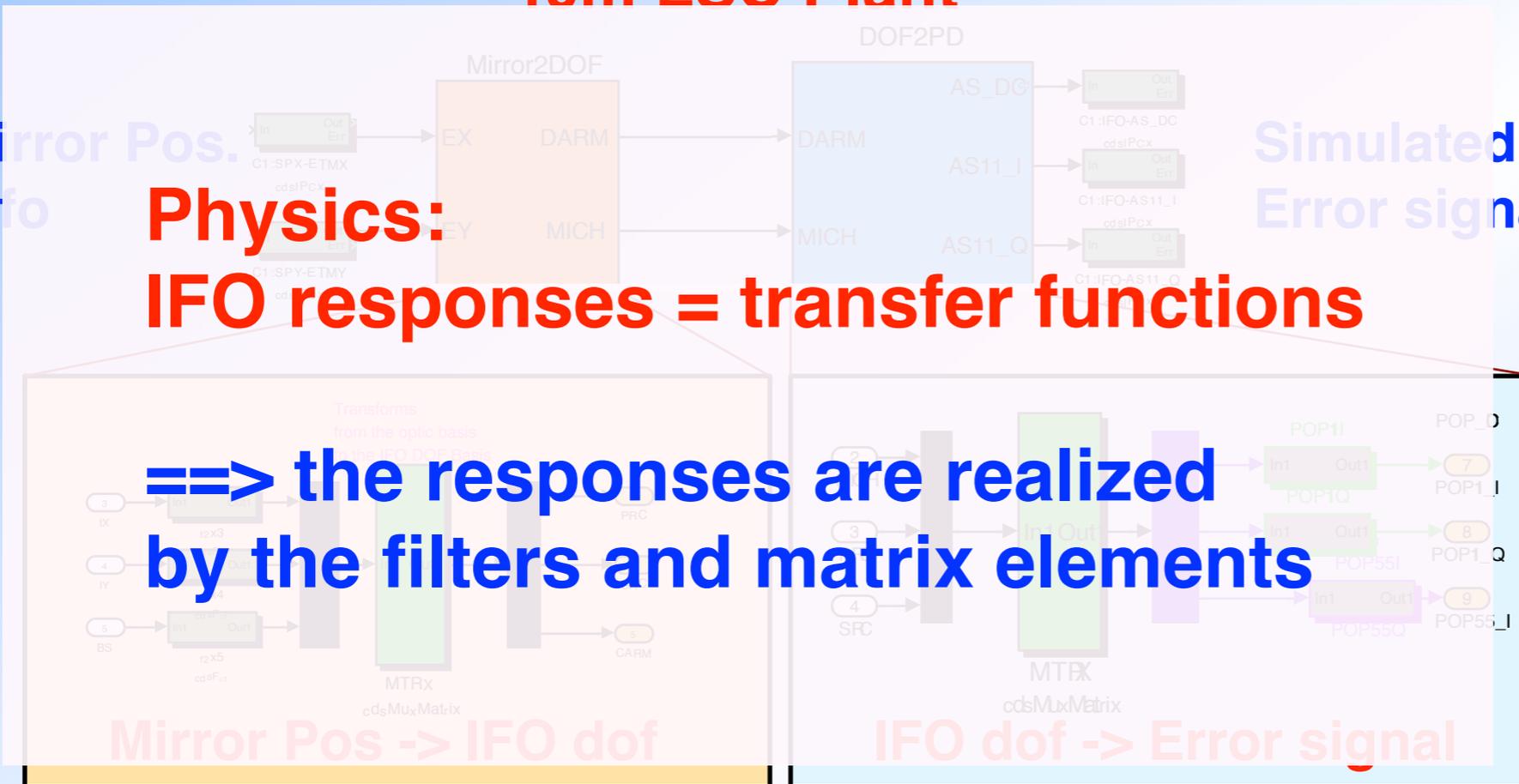
IFO responses = transfer functions

Simulated
Error signal

**==> the responses are realized
by the filters and matrix elements**

Mirror Pos -> IFO dof

IFO dof -> Error signal



Future Plans

- More Noise!

- Add thermal noise, shot noise, radiation pressure
- Improve seismic, add option to use real seismometer data
- Add earthquake stops (to prevent runaway suspensions)

- Finish IFO Plant:

- Allow for the suspension models to talk to each other
- Demonstrate near resonance conditions can be simulated

- Library:

- Yesterday Alex finished a RCG recursive library parser
- Build a library of all 40m control models
- Allows changes to controls to propagate to simulated plant

Just checkout and build

- Statespace RCG block:

- Allow for Statespace simulation in addition to using filters
- Allow for import from currently existing SS models of Quads

Summary

- Simulated plant:
an IFO emulator for commissioning
Realized by the digital control system itself
Could run in parallel on the IFO hardware
- Enables to run IFOs only with the SW
Will make the commissioning easier & faster
Will make the noise hunting easier & faster
Will help the DA development / tests
- Basic suspension simulation at the 40m
Will continue to add noise and improve the plant
- Looking aLIGO quadruple pendulum models
Planning for when we apply these to aLIGO
- Thanks to many people for their contributions:
Rolf Bork, Alex Ivanov, Valera Frolov, Chris Wipf