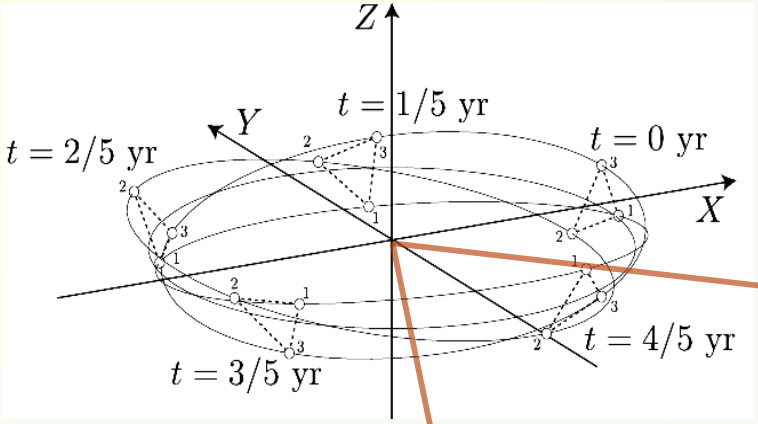
5th LISA Symposium,
July 12-15, 2004

LISA data-analysis tools

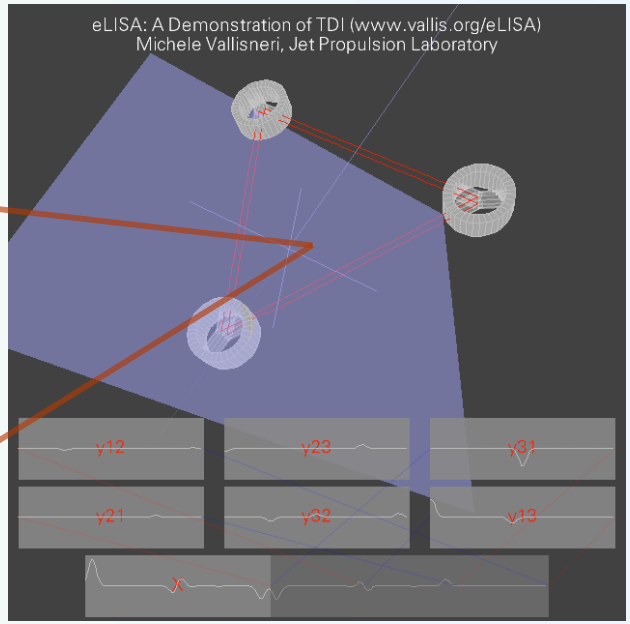
Michele Vallisneri

JPL
CALIFORNIA INSTITUTE OF TECHNOLOGY 1891

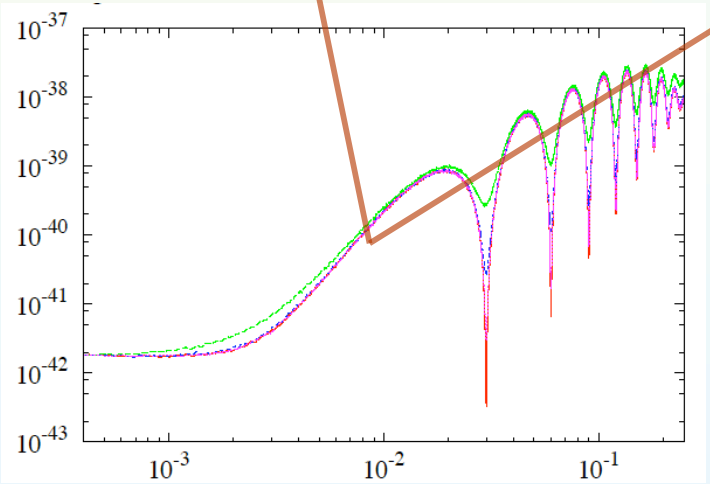
The LISA response



ORBITAL MODULATION



TDI RESPONSE



NOISE SUBTRACTION

Why simulations?

- The LISA response is very complex
- Honest time-domain simulations can:
  - Take over where analytical treatments become unwieldy
  - Increase our trust in analytical insights
  - Point out the unexpected
- Uses of simulation:
  - Development and testing of data-analysis schemes and algorithms
  - Performance characterization and architecture tradeoffs
  - Interface between scientific and technical mission requirements
  - Time-domain studies of noise and vetos
- Two simulators: LISA Simulator (Cornish and Rubbo) Synthetic LISA (Vallisneri and Armstrong)
  - Mutual check (different formalisms for GW response)
  - Slightly different focus (data analysis vs. performance characterization)
  - Compatible conventions, common input format (MLDF)

THE LISA SIMULATOR

Neil Cornish and Louis Rubbo, PRD 67, 022001 (2003)
http://www.physics.montana.edu/lisa

- Conceived as interface between source simulation and data analysis.
  Inputs time series of GW strain in solar system baricenter:
  outputs TDI responses (modified X, Y, Z)
- Computes LISA phase delays by integrating GW strain along photon paths

$$\mathbf{H}(a,b) \equiv \int_b^a \mathbf{h}(\xi)\, d\xi \qquad \delta\ell_{ij}(t_i) = \frac{1}{2}\frac{\hat{r}_{ij}(t_i) \otimes \hat{r}_{ij}(t_i)}{1+\hat{n}\cdot\hat{r}_{ij}(t_i)} : \mathbf{H}(\xi_j,\xi_i)$$

- Models LISA acceleration noise and shot noise (assume Gaussian, white, uncorrelated) → sensitivity if TDI successful
- Includes eccentric LISA orbits, laser-beam look ahead (→ Sagnac delays and flexing of armlengths)
- Has modular design for user upgrades (e.g., sophisticated orbit modules) and future enhancements (e.g., more detailed noise models, additional TDI observables)

THE LISA SIMULATOR

The LISA Simulator is file/shell driven:

Work flow

Modify LISAconstants.h for LISA orbit, cadence, number of samples
→ compile

Modify GW strain executable for source parameters
→ compile

Compute photon vectors, vertex TDI times
→ **geometry files**

Generate proof-mass and shot noise series
→ **noise files**

Generate (or provide) GW strain time series
→ **strain files**

Compute **TDI noises**

Compute **TDI signals**

THE LISA SIMULATOR

The LISA Simulator is file/shell driven:

Work flow

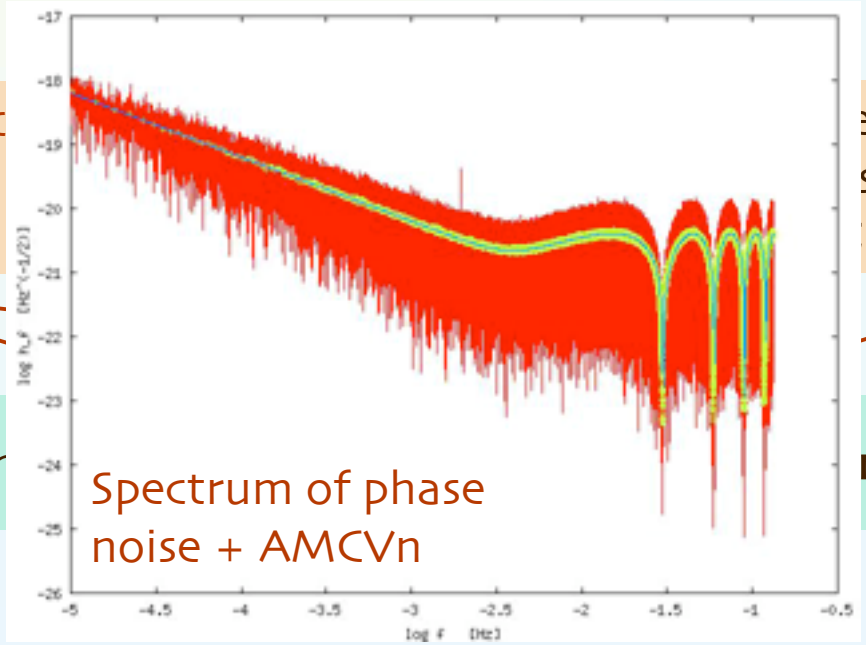
Modify LISAconstants.h for LISA orbit, cadence, number of samples
→ compile

Modify GW strain executable for source parameters
→ compile

Compute photon vec… vertex TDI times
→ geometry files

…erate (or provide) …strain time series …rain files

Com… …nals



Use cases

- Rubbo, Cornish, Poujade, PRD 69, 082003 (2004): validate rigid adiabatic approximation, evaluate long-wavelength approximation
- Installed and used at:
  - Montana State (generating data to test data-analysis algorithms)
  - U. Colorado, Boulder (implementing different low-frequency noise spectra)
  - GSFC (processing GW strain waveforms for MLDA)
  - Caltech (processing SMBH and EMRI signals for MLDA)
  - U. Birmingham (comparing with analytic response expressions)
  - MIT (processing EMRI waveforms for MLDA)
  - UT Brownsville (preparing for LISA data-analysis investigations)
- Version 3.0 with full set of TDI variables, improved interpolation routines, and new data format will be released later this year

Synthetic LISA

M. Vallisneri & J. Armstrong

- Simulates the LISA fundamental noises and GW response at the interface of scientific and technical mission requirements
  - Generates synthetic time series of the LISA fundamental noises in the TDI variables
  - Provides a streamlined module to compute the TDI responses to GW signals
- Includes a full model of the LISA science process (shearing LISA motion, causal light propagation, second-generation TDI, laser-noise subtraction, phase locking)
- Is modular: contains standard noise and GW objects, but it's easy to implement or load new ones
- Is steerable, user-friendly, and extensible (C++, Python, XML), to allow easy interfacing to extended modeling and data-analysis applications
- Is award-winning (Space Act) public-domain software
- Try for yourself: www.vallis.org/syntheticlisa
  www.openchannelfoundation.org (soon)

A SynthLISA Block Diagram

GW sources

for plane waves, work from k, $h_+(t)$, $h_x(t)$ at SSB

LISA noises

laser freq. fluctuations, proof mass, optical path

LISA geometry

spacecraft positions
→ photon propagation
→ armlengths

GWs → instantaneous laser Doppler shifts

$$y_{slr}^{\mathrm{gw}}(t) = \left[1 + \hat{k}\cdot\hat{n}_l(t)\right] \times \left(\Psi_l[t_{\mathrm{send}} - \hat{k}\cdot\vec{p}_s(t_{\mathrm{send}})] - \Psi_l[t - \hat{k}\cdot\vec{p}_r(t)]\right)$$

$$\Psi_l(t') = \frac{\hat{n}_l(t)\cdot\mathbf{h}(t')\cdot\hat{n}_l(t)}{2\left(1 - [\hat{k}\cdot\hat{n}_l(t)]^2\right)}$$

Doppler y_{ij}

inter-spacecraft relative frequency fluctuations

Doppler z_{ij}

intra-spacecraft relative frequency fluctuations

TDI observables

time-delayed combinations of y_{ij} and z_{ij}

laser-noise and optical-bench-noise free

3 independent observables

The Synthetic LISA Package

Implements the LISA science process as a collection of C++ classes

Class LISA

Defines the LISA time-evolving geometry (positions of spacecraft, armlengths)

OriginalLISA: static configuration with fixed (arbitrary) armlengths

ModifiedLISA: stationary configuration, rotating with T=1yr; different cw and ccw armlengths

CircularRotating: spacecraft on circular, inclined orbits; cw/ccw, time-evolving, causal armlengths

EccentricInclined: spacecraft on eccentric, inclined orbits; cw/ccw, time-evolving, causal armlengths

NoisyLISA (use with any **LISA**): adds white noise to armlengths used for TDI delays

...

Class Wave

Defines the position and time evolution of a GW source

SimpleBinary: GW from a physical monochromatic binary

SimpleMonochromatic: simpler parametrization

InterpolateMemory: interpolate user provided buffers for h_+, h_x

...

Class TDI(LISA,Wave)

Return time series of noise and GW TDI observables (builds causal y_{ij}'s; includes first- and second-generation observables)

TDInoise: demonstrates laser-noise subtraction

TDIsignal: causal, validated vs. *LISA Simulator*

TDIfast: cached for multiple sources (Edlund)

The Synthetic LISA Package

Implements the LISA science process as a collection of C++ classes

Class LISA

Defines the LISA time-evolving geometry (positions of spacecraft, armlengths)

OriginalLISA: static configuration with fixed (arbitrary) armlengths

ModifiedLISA: stationary configuration, rotating with T=1yr; different cw and ccw armlengths

CircularRotating: space
inclined orbits; cw/ccw, t
causal armlengths

EccentricInclined: spacecraft on eccentric, inclined orbits; cw/ccw, time-evolving, causal armlengths

NoisyLISA (use with any **LISA**): adds white noise to armlengths used for TDI delays

...

Class Wave

Defines the position and time evolution of a GW source

SimpleBinary: GW from a physical monochromatic binary

SimpleMonochromatic: simpler parametrization

InterpolateMemory: interpolate user provided buffers for h_+, h_x

...

Check the sensitivity of alternate LISA configurations

Class TDI(LISA,Wave)

Return time series of noise and GW TDI observables (builds causal y_{ij}'s; includes first- and second-generation observables)

TDInoise: demonstrates laser-noise subtraction

TDIsignal: causal, validated vs. *LISA Simulator*

TDIfast: cached for multiple sources (Edlund)

The Synthetic LISA Package

Implements the LISA science process as a collection of C++ classes

Class LISA

Defines the LISA time-evolving geometry (positions of spacecraft, armlengths)

OriginalLISA: static configuration with fixed (arbitrary) armlengths

ModifiedLISA: stationary configuration, rotating with T=1yr; different cw and ccw armlengths

CircularRotating: spacecraft on circular, inclined orbits; cw/ccw, time-evolving, causal armlengths

EccentricInclined: spacecraft on eccentric, inclined orbits; cw/ccw, time-evolving, causal armlengths

NoisyLISA (use with any **LISA**): adds white noise to armlengths used for TDI delays

...

Class Wave

Demonstrate laser-noise subtraction

- 1st-generation TDI
- modified TDI
- 2nd-generation TDI
- degradation of subtraction for imperfect knowledge of arms

...

Class TDI(LISA,Wave)

Return time series of noise and GW TDI observables (builds causal y_{ij}'s; includes first- and second-generation observables)

TDInoise: demonstrates laser-noise subtraction

TDIsignal: causal, validated vs. *LISA Simulator*

TDIfast: cached for multiple sources (Edlund)

The Synthetic LISA Package

Implements the LISA science process as a collection of C++ classes

Class LISA

Defines the LISA time-evolving geometry

Produce synthetic time series to test data-analysis algorithms

(arbitrary) armlengths

ModifiedLISA: stationary configuration, rotating with T=1yr; different cw and ccw armlengths

CircularRotating: spacecraft on circular, inclined orbits, cw/ccw, time-evolving, causal armlengths

EccentricInclined: spacecraft on eccentric, inclined orbits; cw/ccw, time-evolving, causal armlengths

NoisyLISA (use with any **LISA**): adds white noise to armlengths used for TDI delays

...

Class Wave

Defines the position and time evolution of a GW source

SimpleBinary: GW from a physical monochromatic binary

SimpleMonochromatic: simpler parametrization

InterpolateMemory: interpolate user provided buffers for h_+, h_x

...

Class TDI(LISA,Wave)

Return time series of noise and GW TDI observables (builds causal y_{ij}'s; includes first- and second-generation observables)

TDInoise: demonstrates laser-noise subtraction

TDIsignal: causal, validated vs. *LISA Simulator*

TDIfast: cached for multiple sources (Edlund)

Running Synthetic LISA

Synthetic LISA is steered with simple Python scripts
What can we do with ten lines of code?

Import SynthLISA library

Create LISA geometry object

Create noise object

Create GW object

Create TDI object

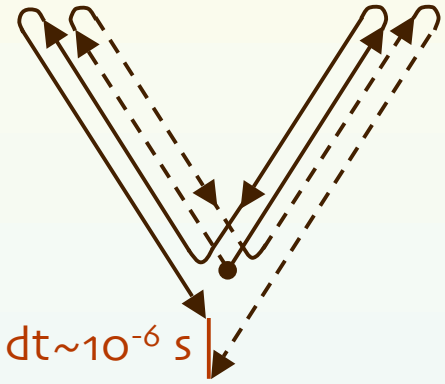
Get X noise and X signal

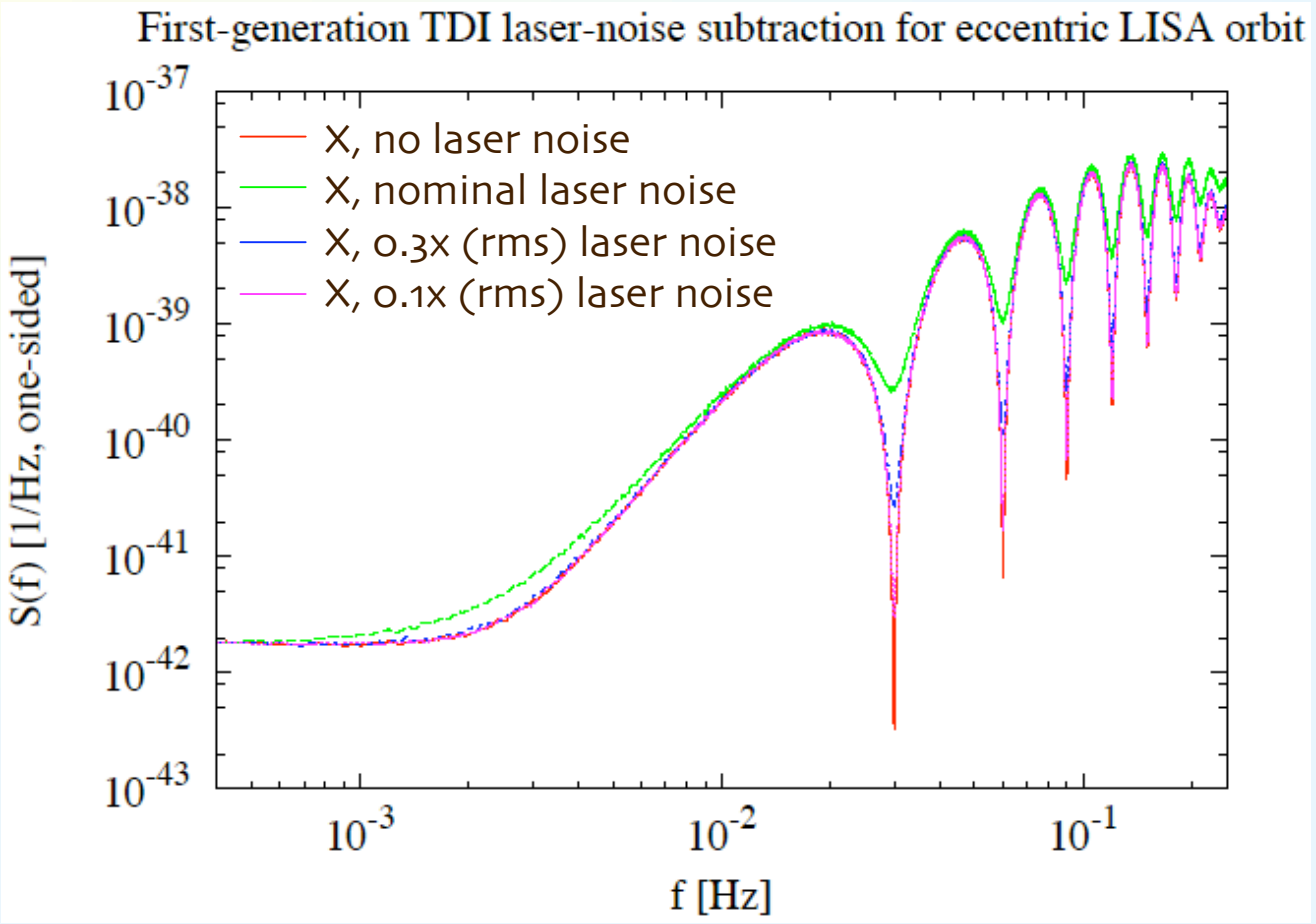
Write spectra to disk

```
1  from lisaswig import *
2  from lisautils import *
                                  Init. LISA pos. and orientation
3  lisa    = EccentricInclined(0.0, 0.0)
                                  Noise sampling rate and spectral density
4  noise   = TDInoise(lisa, 1.0, 2.5e-48,
                      1.0, 1.8e-37, 1.0, 1.1e-26)
                                  Source freq., init. phase, incl.
5  wave    = SimpleBinary(1e-3, 0.0, 0.0,
                          1e-20, 1.57, 0.0, 0.0)
                                  Source ampl., eclipt. lat. & lon., polarization
6  signal  = TDIsignal(lisa, wave)
                                  Requested samples, sampl. rate, observable
7  noiseX  = getobsc(2**16, 1.0, noise.Xm)
8  signalX = getobsc(2**16, 1.0, signal.Xm)
                        Output file      Observable, sampl. rate, avg. periods
9  writearray('noise.txt',  spect(noiseX, 1.0, 64))
10 writearray('signal.txt', spect(signalX,1.0))
```

Case study: laser-noise cancellation for flexing LISA

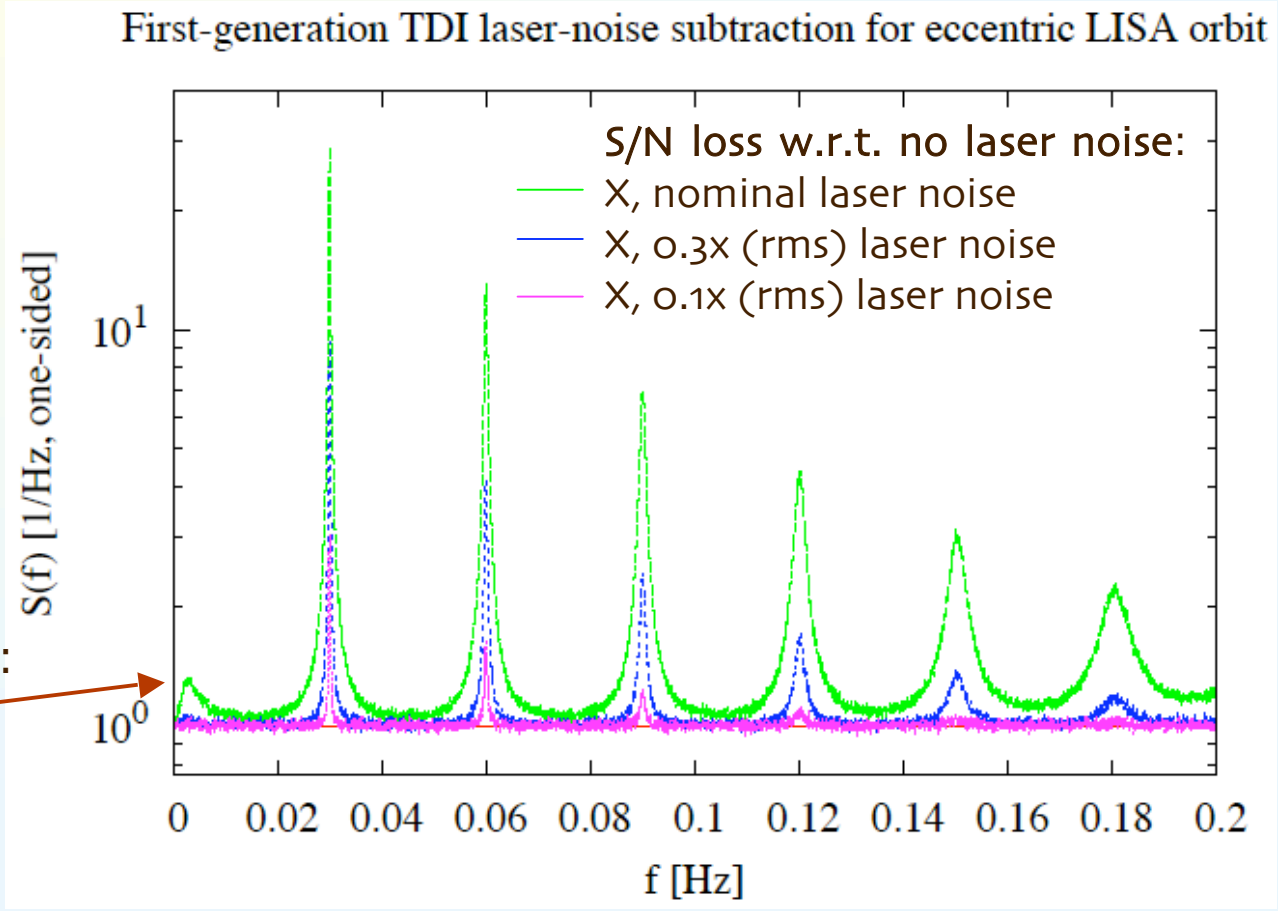
The eccentricity of the LISA orbits induces $\dot{L}_i \sim 10^{-8}$ s/s: the two X beam paths (22′3′3 and 3′322′) become mismatched by $dt \sim 10^{-6}$ s, with a (fractional) laser noise residual $[\dot{C}(f) \times dt]/C(f) \sim 2\pi f \times dt > 10^{-8}$ above 1 mHz (derivative theorem)





Case study: laser-noise cancellation for flexing LISA

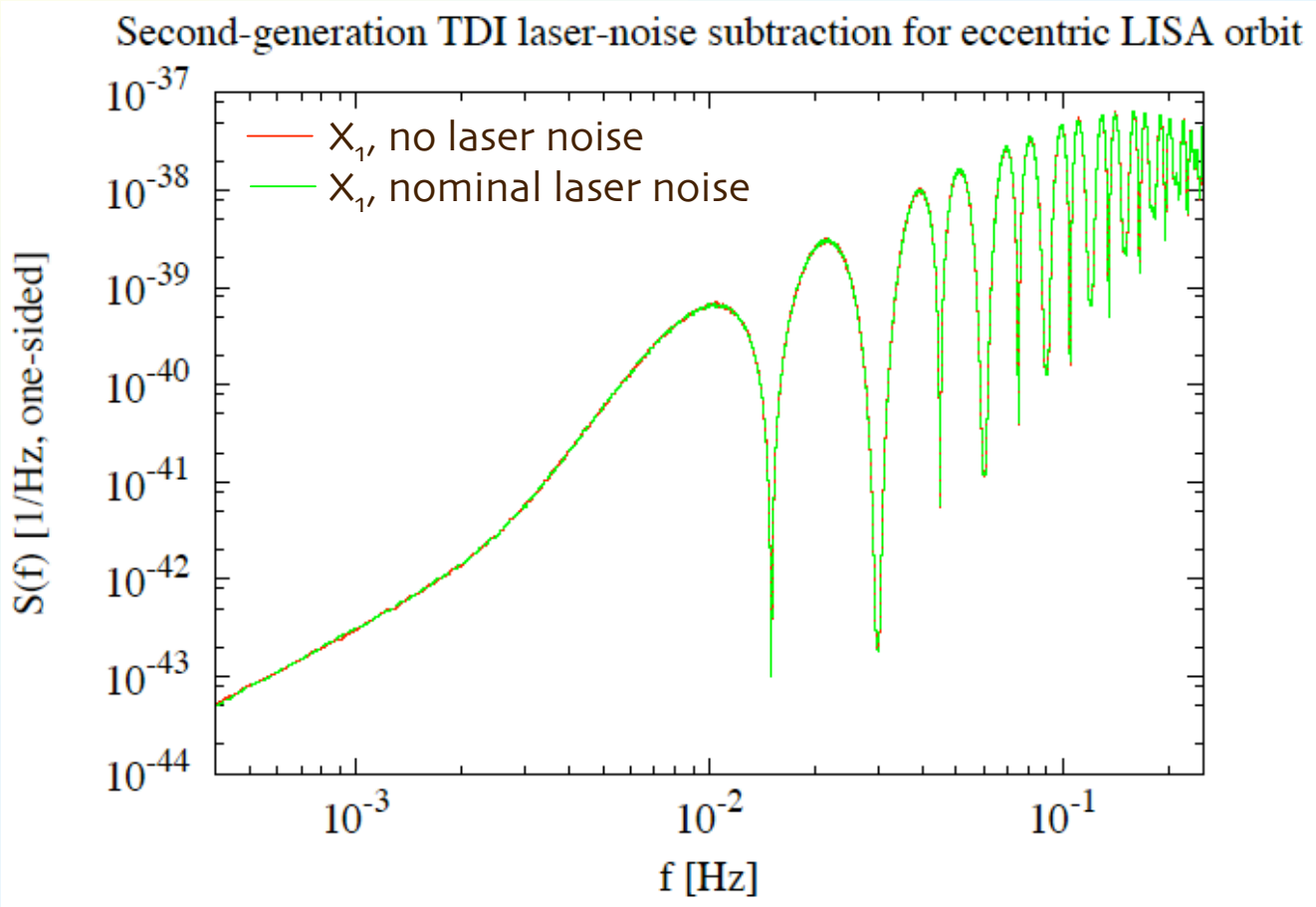
Potential surprise: is second-generation TDI really needed?



note linear scale: this bump spans 10 mHz

Case study: laser-noise cancellation for flexing LISA

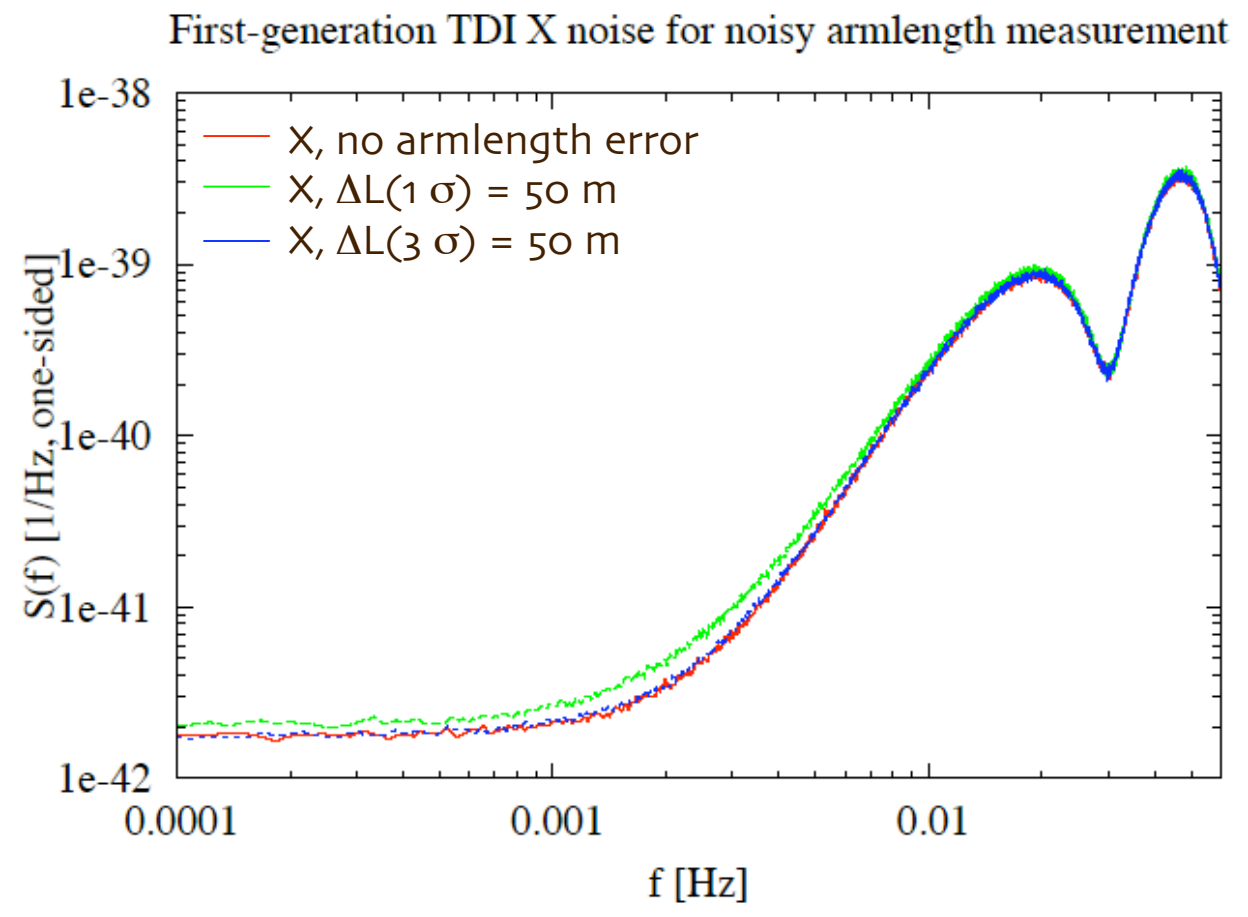
As expected, the second-generation TDI observable X_1 reduces the beam mismatch (between 22′3′33′322′ and 3′322′22′3′3) to dt ~ 10^{-10} s, small enough to cancel nominal laser noise at all frequencies.



Case study: laser-noise cancellation for noisy armlengths

In accordance with analytical estimates, the armlength-measurement tolerance needed for first-generation TDI is ~ 50 m, or 1.7×10^{-7} s (residual laser noise is still present because of flexing).

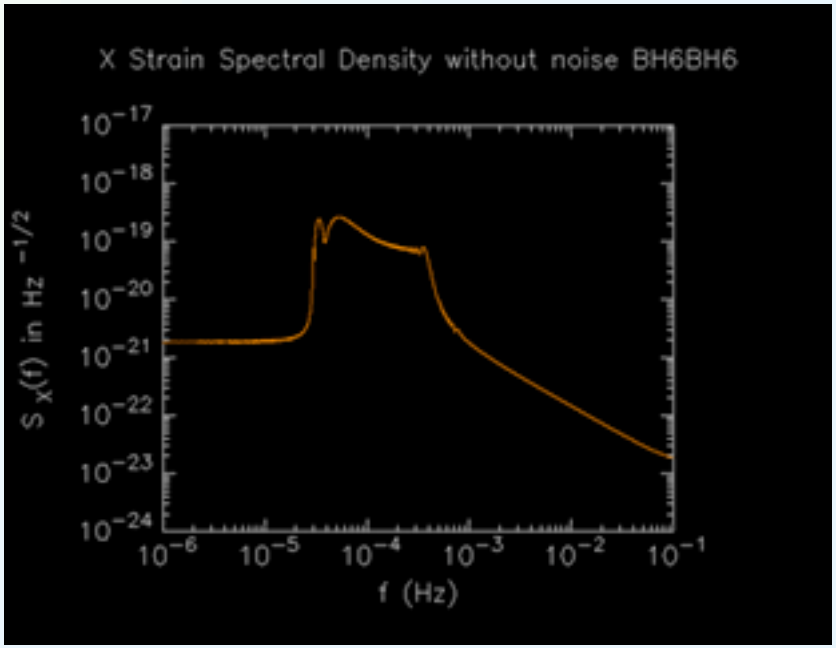
To simulate the armlength error of an orbital-dynamics model updated every 16 s (i.e., the minimum time it takes L to change by one part in 10^8): use white additive noise bandlimited at 1/(2 x 16 s), set total variance to fraction of tolerance



Mock LISA Data Archive

- Hosted at astrogravs.nasa.gov
- Steering Committee: N. Cornish, J. Baker, M. Benacquista, J. Centrella, S. Hughes, S. Larson, M. Vallisneri
- Collect realistic time series of GW strains and of the LISA outputs, for use in developing and testing data-analysis algorithms

- Formulate standard parameter conventions:
  - Where is the source?
  - Whence does LISA start?
  - Which is the plus polarization?
  - What is X?
- As more contributions become available, will develop DB search and indexing capabilities
- Organize mock data challenges



The Mock LISA Data Input Format

```
<?xml version="1.0"?>
<!DOCTYPE XSIL SYSTEM "xsil.dtd">

<XSIL Name="GWPlaneWaveSource">
  <Param Name="EclipticLatitude"   Unit="Degree">40.0</Param>
  <Param Name="EclipticLongitude"  Unit="Degree">180.0</Param>
  <Param Name="SourcePolarization" Unit="Degree">0.0</Param>

  <Time  Name="StartTime" Type="ISO-8601">2004-07-15 06:30:00.0</Time>
  <Param Name="Cadence"   Unit="s">1.0</Param>
  <Param Name="Duration"  Unit="s">65536.0</Param>
```

- An application of XSIL (extensible scientific interchange language; also used as LIGO_LW)
  - Easily parsed by machine and human (web browsers, DBs)
  - Inline or externally linked data
- Read/written by LISA Simulator and Synthetic LISA

Optional parameters go here

```
  <Array Name="hp" Type="double">
    <Dim Name="Length">65536</Dim>
    <Dim Name="Records">1</Dim>

    <Stream Type="Remote"
      Encoding="Bigendian">
      hp.bin</Stream>
  </Array>
</XSIL>
```

5th LISA Symposium,
July 12-15, 2004

LISA data-analysis tools

Michele Vallisneri