NDS Interface for Mathematica

Laser Interferometer Gravitational-wave Observatory
LIGO T990124-D
December 1999, Version 1.0

Daniel Sigg
LIGO Hanford Observatory
P.O. Box 1970 S9-02
Richland, WA

Getting started

Features

The *Network Data Server* (NDS) interface for *Mathematica* supports remote and local access to the data stream of the Laser Interferometer Gravitational-wave Observatory (LIGO). The user can retrieve data from right within a *Mathematica* notebook by importing the *NDS package*. The *NDS package* is build around the *MathLink* interface which allows data exchange through standard network protocols. At the obseravtories a *MathLink* server program is constantly listening to user requests which are then translated into requests for the *Network Data Server*. When the *Network Data Server* completes a request the data is sent back through the *MathLink* interface to the *Mathematica* notebook where it is then accessible to the user for further processing and viewing as a native *Mathematica* data array.

Installation

At the user side the *NDS package* "nds.m" must be installed at a placed where it can be seen by the *Mathematica* notebook. Most conveniently a new directory with name "LIGO" is created in the *Mathematica* subdirectory "Mathematica/AddOns/Applications". After copying the *NDS package* into this new folder the user can load it into the notebook with

```
Needs["LIGO`nds`"]
```

or if you installed the package under the current path load it with

```
<< "./nds.m"
```

Get the package from here

http://www.ligo-wa.caltech.edu/gds/math/Download/nds.m

First steps

To test if the new package the user can request a list of channel names and computes the length of the returned list

```
channels = NDSGetChannel[Site → LHO];
Length[channels]
```

In the above example the site from which to obtain the channel names has been explicitely specified as an option. When obmitted the default site is determined by the variable

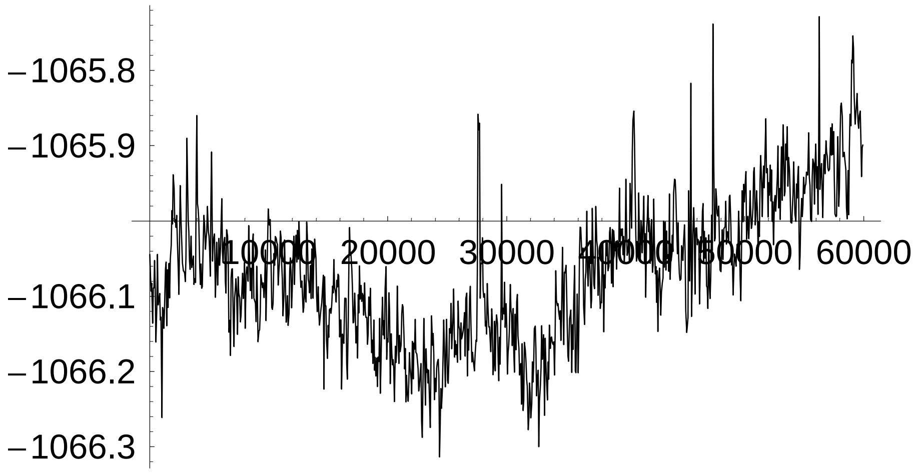
```
Site → $DefaultSite
```

Currently supported site values are

`LHO`	for the LIGO Hanford Observatory
`LLO`	for the LIGO Livingston Observatory

Now get some data from the server and plot it

```
$DefaultSite = LHO;
t0 = {1999, 12, 12, 16, 57, 0};
trend = NDSGetTrend[{"H0:PEM-LVEA_SEISX"},
   t0, 1000 * 60, DataRate → 1. / 60.];
ListPlot[{#⟦1⟧, #⟦3⟧} & /@ (trend⟦1⟧), PlotJoined → True];
```



If we are done, we can close the connection to the network data server. Otherwise, the connection stays open and will be reused if the same site is selected for the next request.

```
NDSClose[]
```

Channel Information

`NDSGetChannel[]`	obtain a list of channel names
`NDSGetChannelInfo[]`	obtain a list of channel information records

`NDSGetChannel` returns a list of string values describing the channel names, whereas `NDSGetChannelInfo` returns a list of channel information records with the following elements:

`ChannelName → name`	Name of the channel
`DataRate → rate`	Data rate of the channel
`InterferometerId → ifo`	Interferometer identification
`DCU → id`	Data collection unit identification
`NDSDataType → dtype`	Data type

Requesting Fast Data

`NDSGetData[c,` t_0`, Δt]`	get raw data of a single channel
`NDSGetData[{`c_1`,` c_2`, …},` t_0`, Δt]`	get raw data of multiple channels

`NDSGetData` requests data from the network data server. Either a single channel name or a list of channel names can be specified as the first argument. The start time t_0 can be specified either as a UTC date or in GPS time. The duration Δ`t` is always specified in seconds.

`{year, month, day, hour, min, sec}`	UTC date
`{GPS sec, nano sec}`	GPS time

The following options are supprted by `NDSGetData`

option	*default value*	*description*
`Site`	`LHO`	location of the data server
`DataRate`	`256`	specifies the data rate
`Heterodyne`	`0`	down–conversion frequency
`DataFormat`	`ColumnFormat`	format of returned array

The rate of retunred data channels has to identical for all requested channels. If the native data rate of a channel is larger than the requested rate, the data is automatically filtered and decimated. If the native data rate of a channel is smaller than the requested rate, the missing data points are filled in using linear interpolation. If the heterodyne frequency is non-zero, the data is down-converted first and the returned data array consists of complex numbers representing both the in-phase and quad-phase signal. The returned array always contains an extra channel describing the time of the data points. The format of the returned data is a two dimensional array either in column or row layout. In column format the data is organized as a list of channel data with the first column representing the time, the second column representing the first channel, and so on. The row format is the transposed of the column format with the data organized as a list of measurement points.

Requesting Trend Data

`NDSGetTrend[c, t0, Δt]`	get trend data of a single channel
`NDSGetTrend[{c1, c2, …}, t0, Δt]`	get trend data of multiple channels

`NDSGetTrend` requests trend data from the network data server. Either a single channel name or a list of channel names can be specified as the first argument. The start time t_0 can be specified either as a UTC date or in GPS seconds. The duration Δt is always specified in seconds. The data is returned as a three dimensional array describing a list of channels. Each channel then descibes a list of data points. Each data point is represented by

$\{t, n, \frac{1}{n} \sum_{i=1}^{n} x_i, \sigma_x, \min\{x_i\}, \max\{x_i\}\}$	Trend data point

The following options are supprted by `NDSGetTrend`

option	default value	description
`Site`	`LHO`	location of the data server
`DataRate`	`1`	specifies the data rate

Trend data is stored by the network data server at both the one second and the one minute rate. Therefore the requested data rate must be smaller than one. For rates between, $\frac{1}{60} < f \leq 1$, the one seconnd trend information is used, whereas for rate smaller than, $f < \frac{1}{60}$, the rate is rounded to the closest minute interval and the one minute trend information is used.

Examples

Example 1: Looking at seismic activities

We first load some graphics packages and setup a few common plot options.

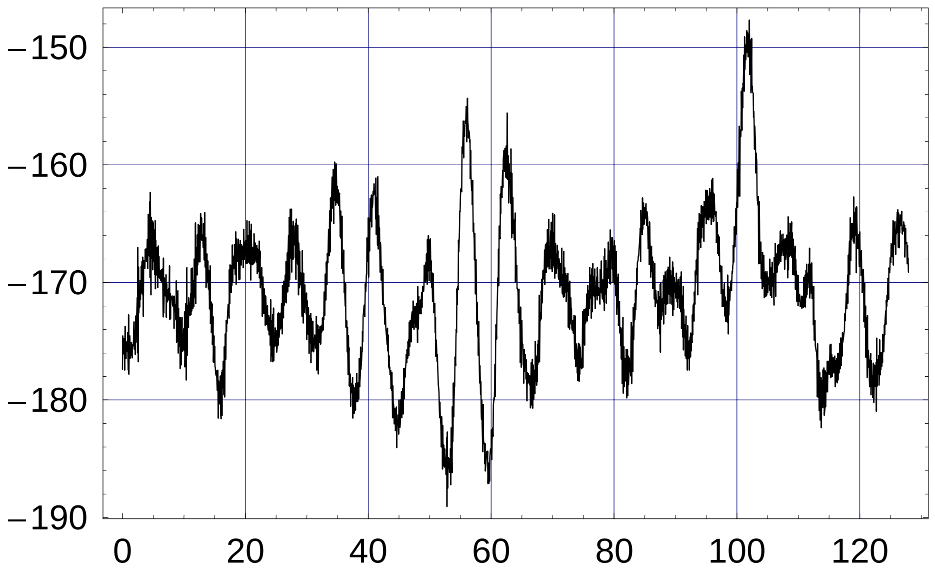
```
Needs["Graphics`Graphics`"];
Needs["Graphics`MultipleListPlot`"];
Needs["Statistics`DescriptiveStatistics`"];
$TextStyle = {FontFamily → "Helvetica", FontSize → 13};
plotopt = Sequence[Frame → True,
   GridLines → Automatic, PlotJoined → True];
```

To take a look at the microseismic peak we retrieve a stretch of data from a seismometer in the mid station covering an interval of 128 seconds at a sampling rate of 16 Hz (after decimation). This request will take a while...

```
t0 = ToDate[FromDate[Date[0]] - 3600];
data = NDSGetData[{"H0:PEM-MX_SEISZ"}, t0, 128,
   DataRate → 16, Heterodyne → 0, DataFormat → RowFormat];
```

First we take a look at the time trace

```
ListPlot[data, plotopt];
```



After subtracting the mean value from the time series we compute the Fourier series and estimate the power spectral density (no windowing function)

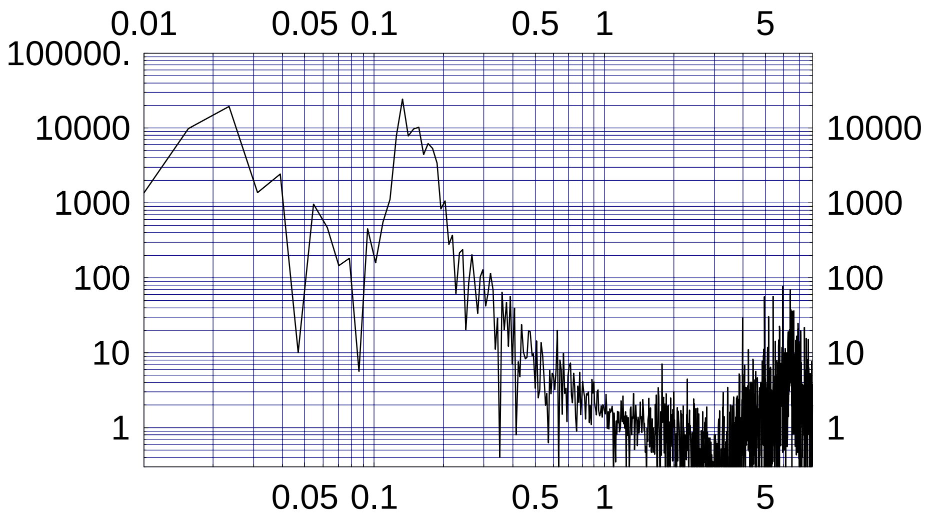
```
ts = Transpose[data]⟦2⟧;
ts = ts - Mean[ts];
```

```
PowerSpectrum[l_List] :=
 With[{len = Round[Length[l]/2]}, Abs[Take[l, len]]^2 +
   Abs[Prepend[Reverse[Take[l, -len + 1]], 0]]^2]
```

```
ps = PowerSpectrum[Fourier[ts]];
ps = Transpose[{Table[i / 128., {i, 0, Length[ps] - 1}], ps}];
```

Plotting the result we clearly see the peak at the microseismic frequency around 0.1–0.2 Hz.

```
LogLogListPlot[Drop[ps, 1], plotopt,
  PlotRange → {{0.01, 8}, {0.3, 100000}}];
```



Example 2: Investigating an earthquake

There was an magnitude 7 earthquake in Turkey on November 11, 1999, 16:57:20 UTC. The one-minute trend data of the seismometers in the LVEA and the mid stations can be retrived with

```
t0 = {1999, 11, 12, 16, 57, 0};
trend =
  NDSGetTrend[{"H0:PEM-LVEA_SEISX", "H0:PEM-LVEA_SEISY",
    "H0:PEM-LVEA_SEISZ", "H0:PEM-MX_SEISX",
    "H0:PEM-MX_SEISY", "H0:PEM-MX_SEISZ", "H0:PEM-MY_SEISX",
    "H0:PEM-MY_SEISY", "H0:PEM-MY_SEISZ"},
   t0, 10000, DataRate → 1. / 60.];
trend = NDSTrendEliminateInvalid[trend];
```

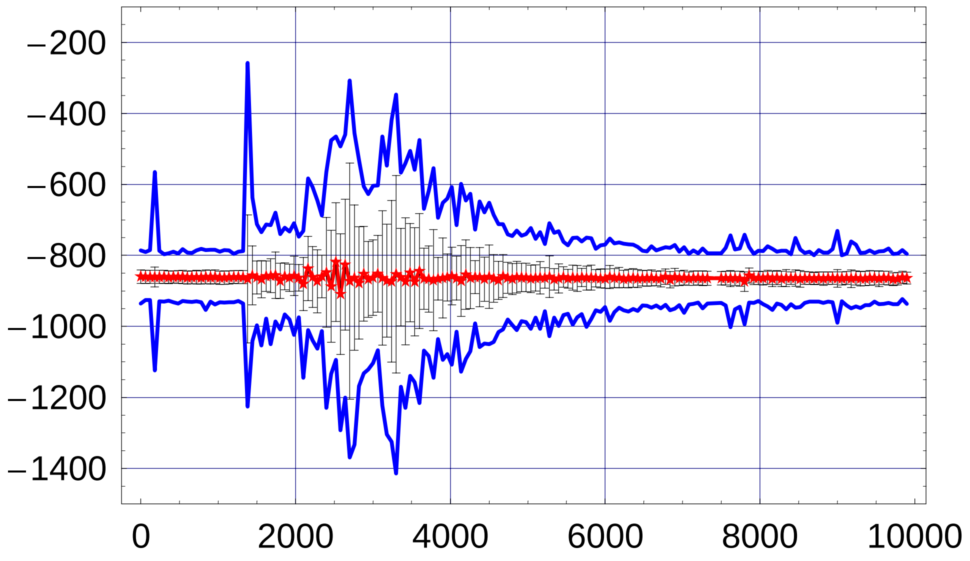
where invalid points have been eliminated. Setting up some plot options first

```
Needs["Graphics`MultipleListPlot`"];
```

```
$TextStyle = {FontFamily → "Helvetica", FontSize → 13};
plotopt = Sequence[Frame → True,
  GridLines → Automatic, PlotJoined → True,
  PlotStyle → {{Thickness[0.005], RGBColor[1, 0, 0]},
    {Thickness[0.005], RGBColor[0, 0, 1]},
    {Thickness[0.005], RGBColor[0, 0, 1]}},
  SymbolStyle → {RGBColor[1, 0, 0]},
SymbolShape → {PlotSymbol[Star], None, None}];
```

one can plot the time series around the time of the earthquake. The red line represents the mean readout of the seismomter, the blue lines indicate the minimum and maximum values at each minute, and the error bars represent the standard deviation of the signal.

```
trace = 1;
MultipleListPlot[
  {{#[[1]], #[[3]]}, ErrorBar[#[[4]]]} & /@ (trend[[trace]]),
  {#[[1]], #[[5]]} & /@ (trend[[trace]]),
  {#[[1]], #[[6]]} & /@ (trend[[trace]]),
  PlotRange → {-1500, -100}, plotopt];
```



Example 3: Compute the total data rate

We define some functions to extract the bytes per second value

```
bytes[n_Integer] :=
 Switch[n, 1, 2,  2, 4,  3, 8,  4, 4,  5, 8,  _, 0]
BPS[rate_Integer, dtype_Integer] := rate * bytes[dtype]
```

Now we get the channel information

```
info = NDSGetChannelInfo[Site → LHO];
```

and compute the data rate

```
Plus @@ ((BPS[DataRate, NDSDataType] /. #) & /@ info) / 1024.^2
 "MB/s"
```

```
17.8282 MB/s
```

Oops! We mistakenly included all digital test points. Since digital test points belong to either Data Collection Units 13 to 16, or 25 to 28, we get a better estimate of the effective data rate of the data acqusition system by using

```
DCUYes[dcu_Integer] :=
 Which[13 ≤ dcu ≤ 16, 0, 25 ≤ dcu ≤ 28, 0, True,  1]
BPS[rate_Integer, dtype_Integer,  dcu_Integer] :=
 rate * bytes[dtype] * DCUYes[dcu]
```

```
Plus @@ ((BPS[DataRate, NDSDataType, DCU] /. #) & /@ info) /
  1024^2. "MB/s"
```

```
5.20317 MB/s
```

Reference

■ ChannelName

`ChannelName` is an options used by **`NDSGetChannelInfo`** to return the channel name.

■ ColumnFormat

`ColumnFormat` is an option value specify the format of the returned data. See **`DataFormat`**.

■ DataFormat

The option **`DataFormat`** specifies the format of the returned data array. Possible values are **`RowFormat`** or **`ColumnFormat`**. In row format the data is returned as a list of n-tuples representing the time, the value of the first channel, the value of the second channel, etc. In column format the data is returned as a list of channels data arrays. The first data array always describes the time.

■ DataRate

`DataRate` is an option to specify the sampling rate of the returned data array. The value is specified in units of Hz.

■ DCU

`DCU` is an options used by **`NDSGetChannelInfo`** to return the DCU information.

■ Heterodyne

The **`Heterodyne`** option is used to down-convert the data by the specified rate before it is returned. The value is specified in units of Hz. The returned data values will be complex.

■ InterferometerId

`InterferometerId` is an options used by **`NDSGetChannelInfo`** to return the interferometer identification number.

■ LHO

The option value **`LHO`** is used to represente the site of the LIGO Hanford Observatory. See **`Site`**.

■ **LLO**

The option value **`LHO`** is used to represente the site of the LIGO Hanford Observatory. See **`Site`**.

■ **NDSClose**

`NDSClose` terminates the connection to the network data server.

■ **NDSDataType**

`NDSDataType` is an options used by **`NDSGetChannelInfo`** to return the native data type.

■ **NDSGetChannel**

`NDSGetChannel[]` obtains a list of all currently available channel names. The following options can be given:

`Site`	**`$DefaultSite`**	specifies the location of the NDS server.

■ **NDSGetChannelInfo**

`NDSGetChannelInfo[]` obtains a list of all currently available channel names and the information assciated with them. Each channel is represented by a list with the following elements:

```
{ChannelName→"channel", DataRate→rate, InterferometerId→ifo,
 DCU→id, NDSDataType→dtype}.
```

The data rate is specified in units of Hz; the interferometer number is either 0 for the 4k intereformeter or 1 for the 2k; the DCU id is the idendification number of data collection unit which collected the channel; and the data type is one of the following: 1 – short integer (16 bit), 2 – integer (32 bit), 3 – long integer (64 bit), 4 – single precision floating point number (32 bit), and 5 – double precision floating point number (64 bit).

The following options can be given:

`Site`	**`$DefaultSite`**	specifies the location of the NDS server.

■ NDSGetData

`NDSGetData["channel", start, duration]` obtains data for a single channel from the network data server.

`NDSGetData[{"chn1", "chn2", ...}, start, duration]` obtains data for multiple channels from the network data server.

The following options can be given:

`DataRate`	**`256`**	specifies the sampling rate of the returned data
`DataFormat`	**`ColumnFormat`**	specifies the format of the returned data
`Heterodyne`	**`0`**	specifies the down-conversion frequency
`Site`	**`$DefaultSite`**	specifies the location of the NDS server.

■ NDSGetTrend

`NDSGetTrend["channel", start, duration]` obtains trend data for a single channel from the network data server.

`NDSGetTrend[{"chn1", "chn2", ...}, start, duration]` obtains trend data for multiple channels from the network data server.

Each returned data point constist of an n-tuple consisting of the time, the number of points, the mean value, the standard deviation, the minimum and the maximum. The returned data format is always in row format. The data rate has to be smaller or equal one Hertz. For rates faster than once a minute the second trend information is used, whereas for rates equal or slower than once the minute trend information is used. The following options can be given:

`DataRate`	**`1`**	specifies the sampling rate of the returned data
`Site`	**`$DefaultSite`**	specifies the location of the NDS server.

■ NDSTrendEliminateInvalid

`NDSTrendEliminateInvalid[]` eliminates invalid data points from a returned trend data array. Invalid data points are recognized by containing an number of points of zero.

■ RowFormat

`RowFormat` is an option value specifying the format of the returned data. See **`DataFormat`**.

■ `Site`

Site is an option which is used to specify the loaction of the data server. Supported values are **`LHO`** or **`LLO`** or a string describing the port number and server name in the format "1234@daniel.ligo-wa.caltech.edu".

■ `$DefaultSite`

`$DefaultSite` is used to specify the default location of the NDS data server.