Data Analysis Applications Group (DAAG) Update: Build_With_Error()

Justin Burruss
2004.03.19

Reminders: Pre-Ops meeting online, survey online

- DIII-D pre-ops meeting now being broadcast via VRVS
  - Also, the DIII-D control room is being broadcast
  - If you need Headphones and/or web cams, contact Gheni (he has a few)
  - Information available in Gheni's presentation from Monday (in case you missed it, it is available, as usual, online)

- If you have not already completed the DAAG User Survey, please do so
  - It's very helpful so we can better understand our users
  - You can skip questions if you want
  - It is anonymous (also if you want)
  - See link from DIII-D Data Page (or David Schissel's original email or the slides from last week)

Store your error bars with **Build_With_Error()**

- MDSplus data can be store along with units, dimensions, and **error**
- Most data in MDSplus already has units & dimensions, but most do not have error bars
- In many cases it's a good idea to store your error bars
  - When you know the underlying raw data has a certain precision
  - When data processing introduces new errors, such as profile fit data
- MDSplus makes it easy to store your error bars using Build_With_Error()

The usage of Build_With_Error() is straightforward

- This example (in TDI) illustrates

```
/* a simple example array of 30 floats */
_x = 0.0 : 2.9 : 0.1

/* an array of error bars for our data, all 0.1 */
_errs = REPLICATE(0.1, 0, 30)

/* associating the errors with the data */
_data = Build_With_Error(_x, _errs)
```

- In real life, your error bars need not be the same for each data point
- You can have errors for all parts of a signal, including the dependent variable and any independent variables

Build_With_Error() can be read easily

- If you are storing errors, but not with Build_With_Error(), then users (and programs) must look in two places for the data and the errors
- Storing errors with the data means you need only look at one node
- This also means that there's no need to invent conventions for storing your error bars
  - e.g. a **:ERRORS** subnode convention
- It also makes it easy to write programs that add error bars to your data since the errors will be stored in a standard way

```
/* extracting errors from a signal _sig */
_errors = Error_Of(_sig)
```