Data Analysis Tools for LISA PathFinder

M Hewitson for the LTP Team
The DA Mission

- Extract maximum possible science from the data
- Analyse a set of pre-defined experiments
  - Experimental Master Plan
    - around 90 runs, >1 experiment per run
- Perform quasi-real-time analysis
  - feed information forward to following experiments
Requirements

- Need a robust and flexible data analysis environment
  - commercial software (aim to reduce testing overhead)
- Easy to use
  - mission scientists need not be programming experts
- A sufficient and well tested set of tools to carry out the planned analyses
- Traceable and reproducible results
- A team of scientists who can use the tools to
DA activities

- Spectral estimations
  - PSD, Coherence, CPSD, TF
- Parameter estimations
  - Signal extraction, model fitting
- Noise budget
- Noise subtraction
  - Time-series processing, filtering
- Calibration
  - Conversion to acceleration
- Simulation/Modelling
  - Dynamic simulations, system templates, analytical models
Analysis Environment

- **Choices:**
  - commercial software: MATLAB
  - accountable, reproducible results: Analysis Objects
  - graphical interface for analysis design and execution: SIMULINK
  - Multi-user concurrent data access: Client/Server system
What’s an AO?

• Don’t want:
  • images, documents, plots, ascii files, references, web pages

- Name
- Numerical data vectors
- Creation date/time
- Additional flags

- Name
- ID number
- Comment
- pipeline file(s)

Analysis Object

- Numerical Data
- Provenance
- Additional meta-data
- Processing history

- creator
- date
- IP address
- Hostname
- Operating System
- software versions

- Name
- Algorithm version
- Parameter list
- Creation date/time
- Input histories
Taking care of history

Intelligent algorithms

traceable results
Introducing LTPDA

- MATLAB Toolbox
- Implements the AO/history concept
- Object-oriented approach to data analysis
- Fully integrated with MATLAB
- Graphical design and execution of data analysis pipelines

~2500 source files
~100,000 lines of code

Welcome to LTPDA Toolbox
Version: 1.01
Release: (R2008a)
Date: 16-06-08
Intelligent algorithms

\[ b = \text{foo}(a, pl) \]

- Output AOs
- Input AOs
- Input parameters

Algorithmic step:
- Input AO
- Input history
- Algorithm history
- Output AO(s)
How history works

Entire processing history of every AO is recorded
%% Reproduce from history
% Write an m-file from AO
ao2m(a4, 'test.m');

function a_out = test

% TEST.M
%
% written by ao2m / $Id: ao2m.m,v 1.11 2007/11/14 16:30:18 ingo Exp $
%
% based on analysis object:
% name: a3 - 10 / ((Data) + (Data)) - (Data)
% provenance: created by hewitson@localhost[192.168.2.104] on MACI/7.5 (R200
% description:
% original m-file:
%
%
a7331931 = ao(plist([param('VALS', [10]) ]));
a7331814 = ao(plist([param('VALS', [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27]) ]));
a7331773 = ao(plist([param('VALS', [10]) ]));
a7331865 = plus(a7331773, a7331814);
a7331960 = minus(a7331865, a7331931);
a_out = a7331960;

=%a4

% END
plot(a11.hist)

hist2dot(a11.hist, 'a11.dot')
AO methods

Spectral
- pwelch
- tfe
- cohere
- cpsd
- fft/ifft
- dft
- lpsd
- ltfe
- lcohere
- lcpsd

Pre-processing
- resample
- downsample
- upsample
- delay
- detrend
- zeropad
- select
- split
- filter
- whiten
- smooth
- find

Operators
- +, -, *, /, .* , .^ , ./ , ^
- abs, transpose
- sin, cos, tan, exp, ln, log, log10
- min, max, mean, median, mode, std, var
- norm, phase, real, imag
- sign, sort, sqrt, sum, uminus

Modelling
- noise generation
- statespace systems
- pole/zero models
% Create AOs
a1 = ao(10);
a2 = ao('foo.xml');

% Add them
a3 = a1+a2;

% Subtract constant
a4 = a3-10;

%% Make LPSD of data
% Window function
w = specwin('BH92', 10);

% parameter list for lpsd
pl = plist('Kdes', 10, 'Kmin', 2, 'Jdes', 100, 'Win', w);

% use lpsd
a3 = lpsd(a1, pl);

% Load data
ain = ao('inputdata.txt');
aout = ao('outputdata.txt');

% Measure transfer function
tf = tfe(ain, aout);

% Plot
iplot(tf)
% design lowpass filter
filt = miir(plist('standard', 'lowpass', 'fc', 0.1, 'fs', 10));

% Filter a1
a1.filter(filt);

% Make PSD
a1.psd

• Work with objects of various classes
• All processing done by class methods
Visualising data

- Since AOs contain more than just data, nice plots are easy to make

>> iplot(a5)
Storing objects

- Store in ASCII format (long-life)
- XML format for all LTPDA objects

```python
p = plist('a', 2, 'b', 'hello')
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<ltpda_object>
  <object shape="1x1" type="plist">
    <property prop_name="name" shape="1x12" type="char">plist object</property>
    <property prop_name="params" shape="1x2" type="param">
      <object shape="1x2" type="param">
        <property prop_name="name" shape="1x1" type="char">A</property>
        <property prop_name="key" shape="1x1" type="char">A</property>
        <property prop_name="val" shape="1x1" type="double">2</property>
      </object>
    </property>
    <property prop_name="created" shape="1x1" type="time">
      <object shape="1x1" type="time">
        <property prop_name="name" shape="1x11" type="char">time-object</property>
        <property prop_name="utc_epoch_milli" shape="1x1" type="double">1204637764923</property>
        <property prop_name="timezone" shape="1x1" type="sun.util.calendar.ZoneInfo">UTC</property>
        <property prop_name="timeformat" shape="1x1" type="timeformat">
          <object shape="1x1" type="timeformat">
            <property prop_name="format_str" shape="1x23" type="char">yyyy-mm-dd HH:MM:SS.FFF</property>
            <property prop_name="format_nr" shape="1x1" type="double">-1</property>
          </object>
        </property>
      </object>
    </property>
  </object>
</ltpda_object>
```
Graphical Analysis

- Drag-and-drop analysis construction
- Use SIMULINK as drawing tool
- Pipeline is then executed by underlying LTPDA functions
- GUI implements no additional analysis functionality
- All class functionality is automatically converted into blocks
Other GUIs
Data Access

- Centralised data storage
- Remote client access to data
- Submit and retrieve objects from within MATLAB

Raw data $\xrightarrow{\text{convert to AO\hspace{1pt}S}}$ MySQL Server $\xrightarrow{\text{MATLAB}}$ $\xrightarrow{\text{MATLAB}}$ $\xrightarrow{\text{MATLAB}}$
Searching for objects

- Need meta data to find objects again
- Multiple tables store meta-data about each object
- Object code (XML) stored in table
- Virtual collections for objects submitted concurrently
### Database: ltpda_test

#### Table: objmeta

<table>
<thead>
<tr>
<th>id</th>
<th>obj_id</th>
<th>obj_type</th>
<th>name</th>
<th>created</th>
<th>version</th>
<th>ip</th>
<th>hostname</th>
<th>os</th>
<th>submitted</th>
<th>experiment_title</th>
<th>experiment_desc</th>
<th>analysis_desc</th>
<th>quantity</th>
<th>additional_comments</th>
<th>object_management</th>
<th>object_searching</th>
<th>object_viewing</th>
<th>repository_administration</th>
<th>full_web_interface_to_repositories</th>
<th>object_management</th>
<th>object_searching</th>
<th>object_viewing</th>
<th>repository_administration</th>
<th>full_web_interface_to_repositories</th>
</tr>
</thead>
<tbody>
<tr>
<td>271</td>
<td>272</td>
<td>ao</td>
<td>sine wave + noise</td>
<td>2008-02-26 00:44</td>
<td>1.0</td>
<td>193.205.193.171</td>
<td>btlbl</td>
<td>PCWIN</td>
<td>2008-02-26 11:08:45</td>
<td>Repository Test from UTN 2008-02-26</td>
<td>Submit/retrieve test # 4874921</td>
<td>Nothing serious, just playing with submit and retrieve</td>
<td>none</td>
<td>M Hewitson</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>272</td>
<td>273</td>
<td>ao</td>
<td>sine wave + noise</td>
<td>2008-02-26 08:50</td>
<td>1.81</td>
<td>193.205.193.171</td>
<td>btlbl</td>
<td>PCWIN</td>
<td>2008-02-26 11:08:53</td>
<td>Repository Test from UTN 2008-02-26</td>
<td>Submit/retrieve test # 4874929</td>
<td>Nothing serious, just playing with submit and retrieve</td>
<td>none</td>
<td>M Hewitson</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>273</td>
<td>274</td>
<td>ao</td>
<td>sine wave + noise</td>
<td>2008-02-26 08:59</td>
<td>1.81</td>
<td>193.205.193.171</td>
<td>btlbl</td>
<td>PCWIN</td>
<td>2008-02-26 11:09:01</td>
<td>Repository Test from UTN 2008-02-26</td>
<td>Submit/retrieve test # 4874937</td>
<td>Nothing serious, just playing with submit and retrieve</td>
<td>none</td>
<td>M Hewitson</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Testing

Results produced by analysis affect mission time-line

must be well tested!

23

Unit Tests

Test basic building blocks

System Tests

Test overall system (pipelines)

Acceptance Tests

Integrate into operations - perform mission

Version 1

3 candidate releases

150 system tests, 4 fails
LTPDA is a MATLAB toolbox that uses an object-oriented approach to data analysis. Analysis Objects are processed through a data analysis pipeline. At each analysis step, a record is kept of exactly what algorithm was applied to which analysis object and with which parameters. In this way, the result of a particular data analysis is one or more AO, with each AO containing the final result as numerical data together with a full processing history of how the result was achieved.

In addition, there is a SIMULINK interface which allows data analysis pipelines to be built using the SIMULINK diagram editor. The resulting diagram is then executed using a special parser which translates the block diagram into a set of LTPDA-compatible commands.

Another SIMULINK interface is also being developed which uses a more direct approach (no parser). This will be merged into the development tree after some additional testing and exploration.

This toolbox is being developed for use in the data analysis of the LISA Pathfinder Mission.