

SDDS: A Modular Toolkit for Accelerator Simulation, Control, and Operation

Michael Borland
Operations Analysis Group
Advanced Photon Source
Argonne National Laboratory

Outline of Presentation

- History, Concept, and Implementation
- Self-describing data
- Self-Describing Data Sets (SDDS) protocol
- SDDS-compliant program toolkits
- Examples from commissioning
- Examples from operations

A Brief History of SDDS

- Originally (1993), we sought to provide general-purpose software for the APS commissioning team:
 - collect and analyze data
 - perform experiments
 - develop control algorithms
- We planned to have high-level applications (HLAs) written based on *algorithms* developed during commissioning.
- The commissioning concept worked so well that it was used to make HLAs directly.
- The same concept works very well for accelerator simulation.
- SDDS used at IPNS, BESSY II, RHIC, and SLAC.

Concept

- Make a system where programs are operators that sequentially transform data files.
- If programs read/write the same type of file, they can be used in any order.
- Make programs completely generic and not application-specific.
- Well-suited to on-the-fly work because no programming is involved.

Examples of the Concept

- Simple lifetime measurement:
acquire | takeLog | polyFit | display
- Robust lifetime measurement:
acquire | takeLog | polyFit | removeOutliers |
polyFit | display
- Beam history analysis:
acquire | FFT | smooth | peakfind | display
- Find the noisiest power supply:
acquire | computeStats | collect | sort | display
- Chromaticity measurement data reduction:
acquire | smooth | peakfind | collapse | polyFit
| removeOutliers | polyFit | display

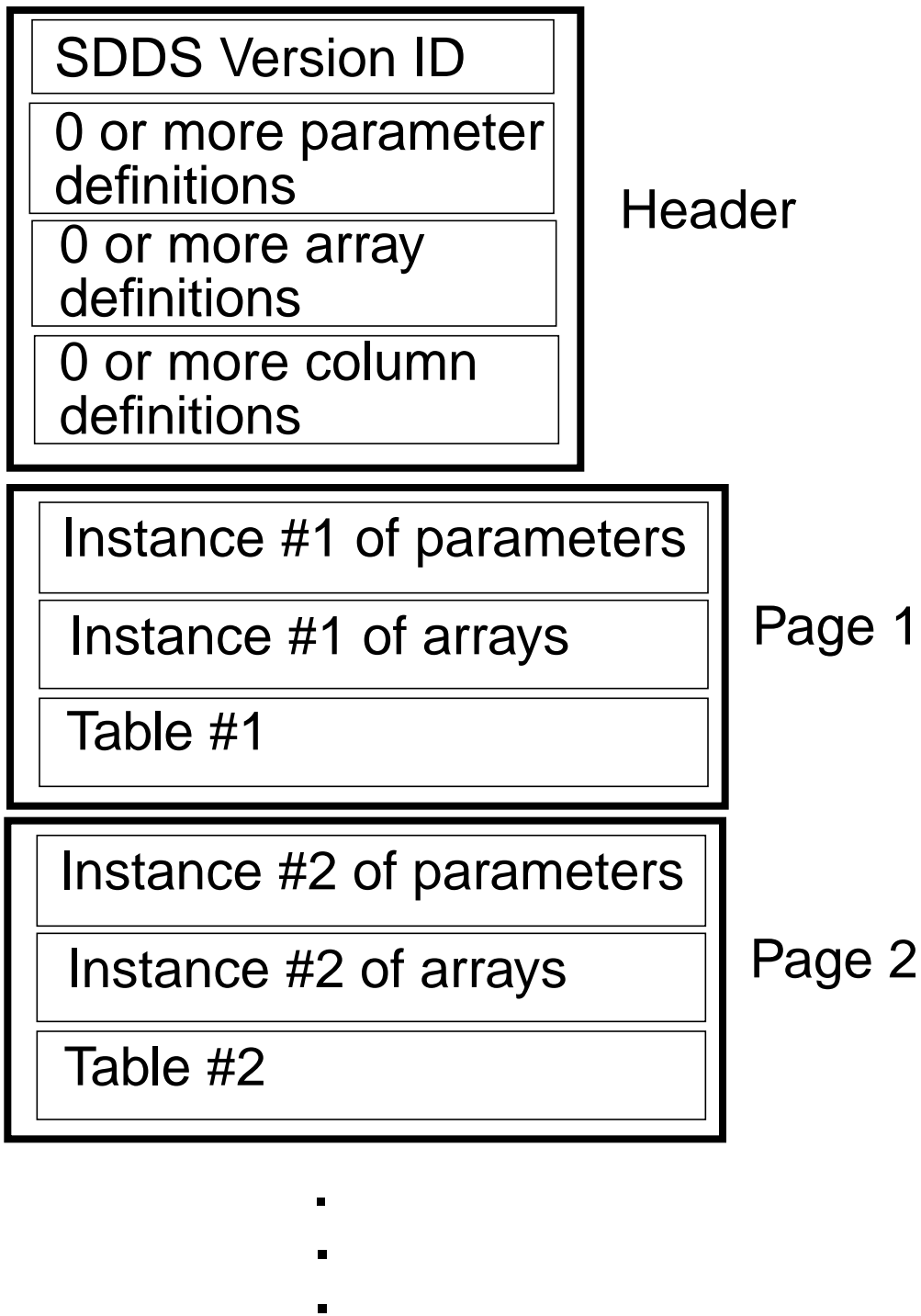
Implementation

- Used a *simple, common* self-describing data protocol for datasets.
- Wrote generic, commandline programs for
 - data collection
 - data analysis
 - graphics
 - process control
- Used pre-existing script languages (e.g., Tcl/Tk) to
 - coordinate programs.
 - record complex sequences for reuse.

What is Self-Describing Data?

- Self-describing (SD) data is identified and accessed by name only.
- SD data files include meta-data about data, e.g., units and data type.
- Advantages:
 - genuinely generic programs possible
 - data elements may be added to files without “breaking” existing programs
 - data tends to be self-documenting
 - source of data (measurement, simulation) is irrelevant

SDDS Data Model



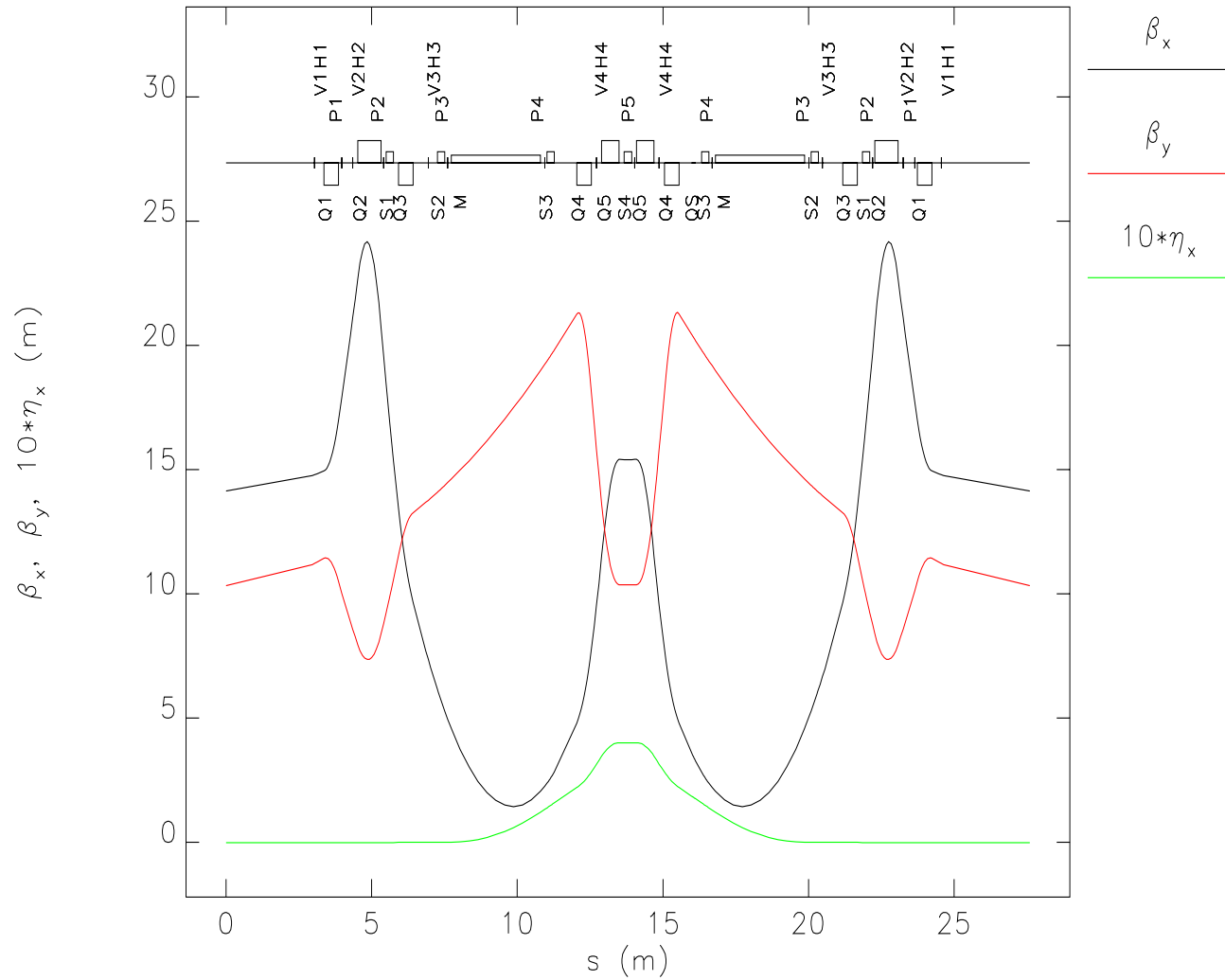
Examples of Control System Data Stored in SDDS Files

- Accelerator backup/restore files.
- Oscilloscope setup data.
- Archival data from continuous machine monitoring.
- Beam dump records.
- Alarm history.
- Magnet conditioning instructions.
- Feedback matrices for energy and trajectory control.
- Response matrices for orbit correction.
- Orbit correction configuration files.
- Waveform data from oscilloscopes and network analyzers.
- Ramp tables for booster power supplies.

Examples of Simulation Data Stored in SDDS Files

- **elegant**—general accelerator modeling
 - lattice functions, matrices, orbits, floor coordinates, etc.
 - orbit correction matrices
 - phase-space coordinate input/output
 - element perturbation input/output
- **spiffe**—electromagnetic PIC simulation
 - EM field maps
 - EM fields at cavity probe points
 - particle snapshots in time and space
 - cavity boundary output
- **shower**—EGS4 interface
 - input particle coordinates
 - shower product phase-space output
 - material properties input
 - radiation dose in distributed materials

APS Twiss Parameters Plotted with 'sddsplot'



Twiss parameters--input: apsSector1.ele lattice: apsSector1.lte

SDDS Toolkit Programs

- SDDS is used by a group of about 70 generic data processing and display programs.
- Most of these “SDDS Toolkit” programs both read and write SDDS files
 - They can be used in sequence.
 - Even simple tools become highly useful when supported by the toolkit.
- About 20 EPICS-specific programs use SDDS.
- Development is decentralized—anyone can add a tool.
- There is no single, large program to maintain.
- All programs are commandline driven and hence scriptable.

SDDS Toolkit Capabilities

- Device-independent graphics.
- Equation evaluation.
- Data winnowing.
- Statistics and histograms.
- Polynomial, exponential, and gaussian fitting.
- Correlation and outlier analysis.
- Matrix operations (e.g., SVD).
- Cross-referencing, sorting, and collation.
- FFTs and digital filtering.
- Protocol conversion to/from SDDS.
- Text printouts of data.

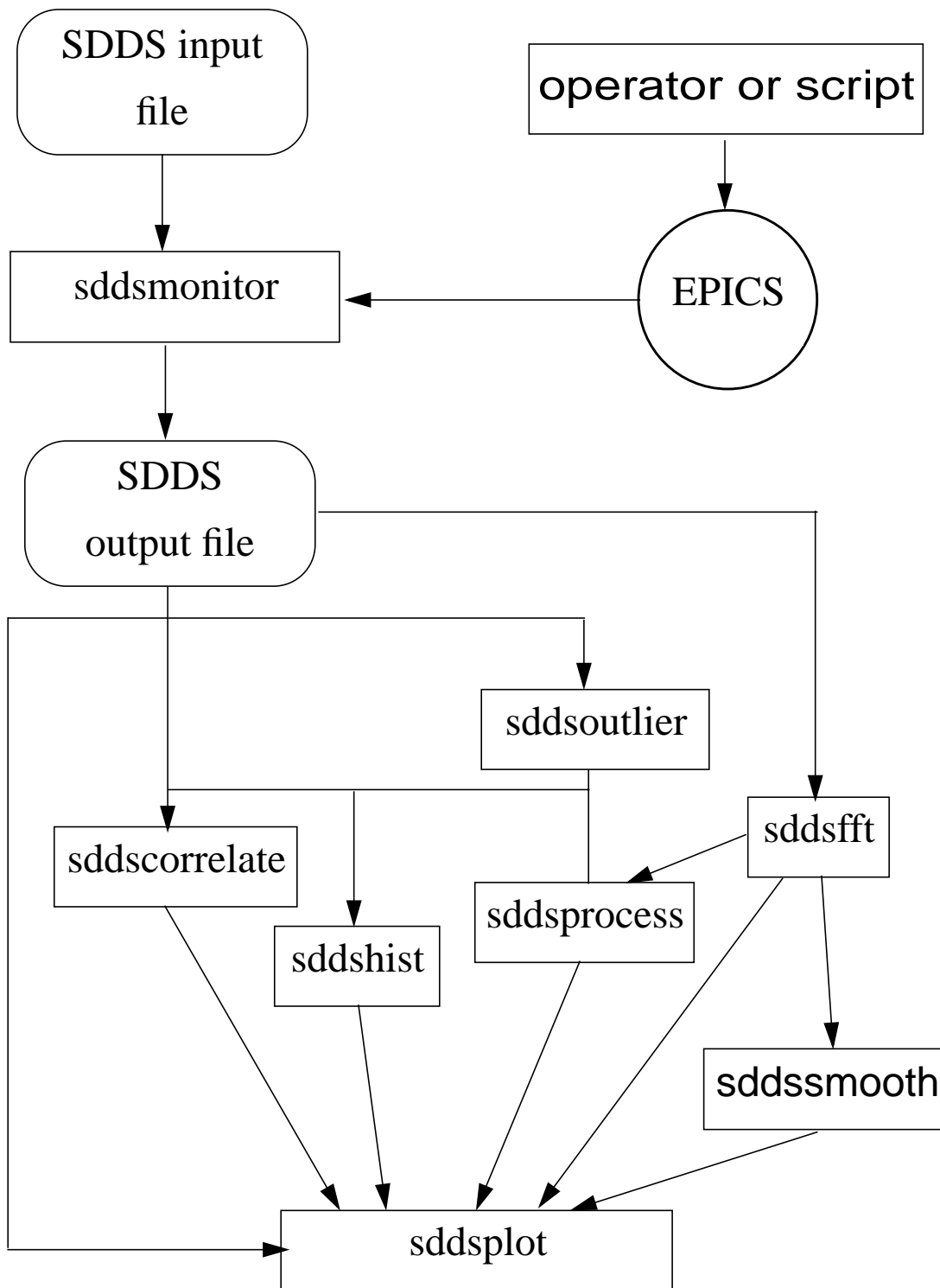
SDDS-Compliant EPICS Programs

- Time-series data collection.
- Time-series statistical data collection.
- Glitch-based data collection.
- Synchronized data collection.
- Alarm data collection.
- Experiment execution.
- Process variable save/restore/ramp.
- Workstation-based feedback.
- Workstation-based feedforward.
- Workstation-based optimization.
- Oscilloscope state save/restore.
- Magnet conditioning setup.

SDDS Monitoring Programs

- Four variants:
 - Logging scalar PV values.
 - Logging statistics of same.
 - Logging scalars grouped as vectors.
 - Logging waveform and scalar PVs.
- Common features
 - SDDS-configured.
 - Acquisition at specified time intervals or on command.
 - Conditional data logging.
 - Erase, append, or generation mode for files.
- Glitch/trigger mode for scalar logging uses a circular buffer for pre- and post-trigger samples.

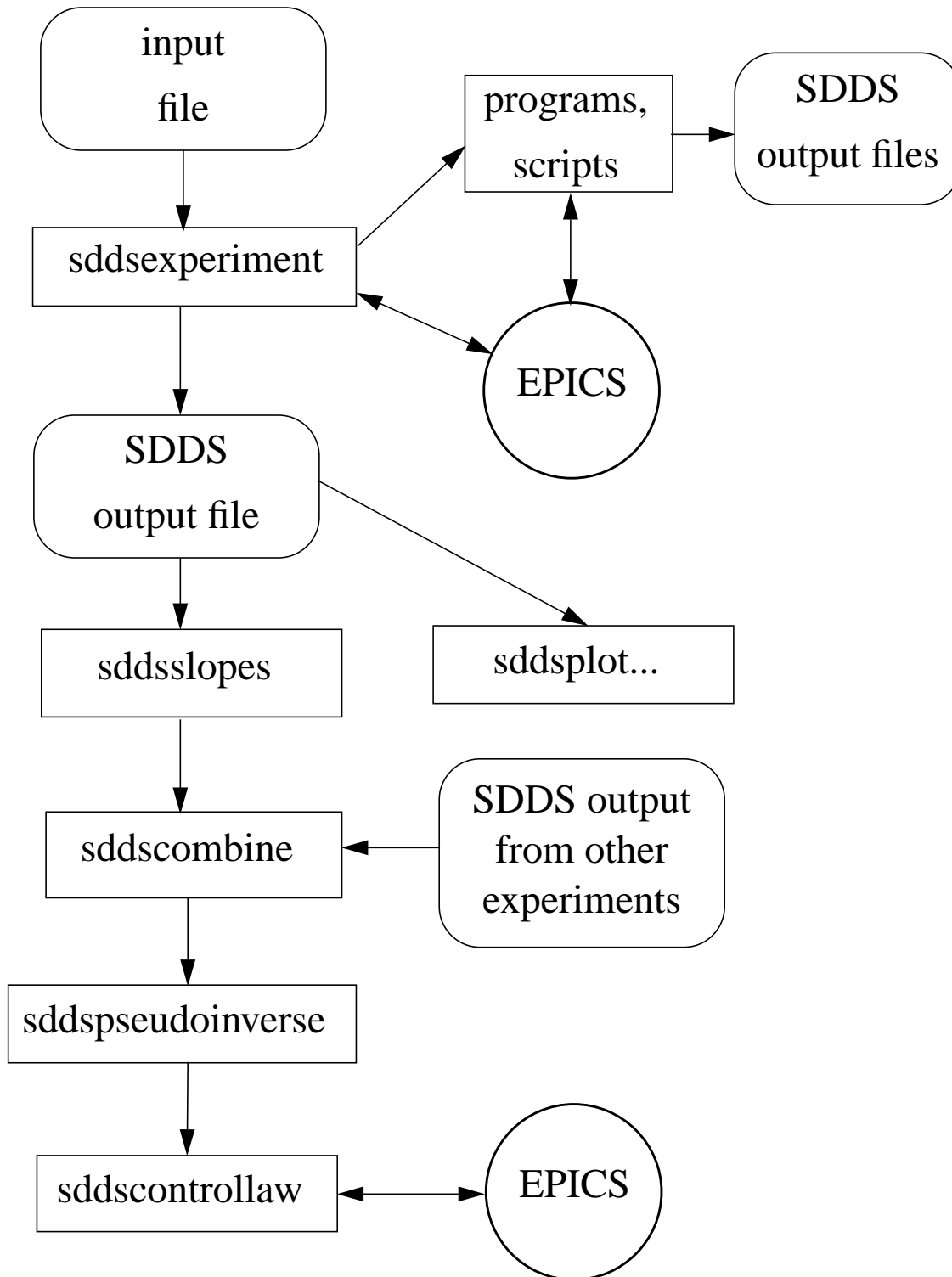
sddsmonitor Application Example



SDDS Experiment Programs

- **sddsexperiment** performs N-dimensional scans and scalar data collection.
- Any number of actuators may be linked to each scan index. Actuator settling time and response testing supported.
- Actuator setpoints may be specified in several ways:
 - Min, max, number of data points.
 - Values from an SDDS file.
 - An equation with one of these supplying input values.
- Actuator variation may be relative or absolute, with optional reset.
- Optionally does averaging, computes error bars, and rejects bad data points.
- External scripts may be linked to scan indices.

sddsexperiment Application Example



sddscontrollaw Overview

- Generic workstation-based feedback program.
- Usually used for integral control:

$$A_{i+1} = A_i - GME_i$$

where A is the vector of actuator values, E is the vector of readbacks, M is the correction matrix, and G is the gain.

- Takes SDDS files for
 - feedback matrix, with actuator and error readback names
 - process variables to test and valid ranges
- Will hold PVs to existing values, to given values (from SDDS file), or to zero.
- Safety features (optional):
 - actuator change limits
 - readback change limits
 - inhibit operation based on PV values
 - “run-control” semaphore system for suspend/resume/abort
 - activity logs

Applications of sddscontrollaw

- Maintains constant energy and trajectory from linac using an experimentally-derived matrix.
- Corrects orbit in PAR.
- Steers/maintains SR orbit using theoretical matrix and orbit despiking filter.
- Maintains SR injection trajectory.
- Adjusts power levels from SR klystrons as beam current changes.
- Regulates power levels from linac klystrons.
- Used by a script that allows on-the-fly creation of one-readback, one-actuator control loops.

Some Commissioning Activities Performed Using SDDS and Scripts

- orbit/trajectory response matrices
- orbit correction algorithm development* (SR)
- dispersion and chromaticity* (SR, PAR)
- automated first-turn steering* (SR)
- beta-functions* (SR)
- dynamic aperture* (SR)
- tune shift with amplitude (SR)
- phase-space tracking (SR)
- tune shift with current (SR)
- BPM intensity dependence* (SR)
- integer tune (SR, booster)
- energy aperture for stored beam* (SR)
- physical aperture search* (SR)
- insertion device effect on orbit and tune* (SR)
- x-ray BPM “pollution” tests (SR)
- septum leakage field using beam (SR, PAR)

- lifetime vs scraper position, bump height* (SR)
- beta-function and dispersion correction* (SR)
- linear coupling reduction (SR)
- kicker bump matching (SR)
- BPM-to-quad offset measurements* (SR)
- search for sources of beam motion (SR)
- power supply ramp correction* (booster)
- automated injection steering (booster)
- longitudinal injection acceptance (PAR)
- bunch length and damping time (PAR)
- rf voltage calibration using beam (PAR)
- kicker waveform shape using beam (PAR)
- automated beam-excited HOM search (PAR)
- energy gain vs input power, frequency, and temperature (linac)
- klystron gain measurements (linac)
- search for cause of energy oscillations (linac)
- emittance measurements and beta-function matching* (linac)

SR Beam History

- The storage ring BPM system has several history buffers that store samples for selected BPMs at rates from 60Hz to 271kHz.
- Data is available in EPICS waveform records, which we download with **sddswmonitor**.
- For example:

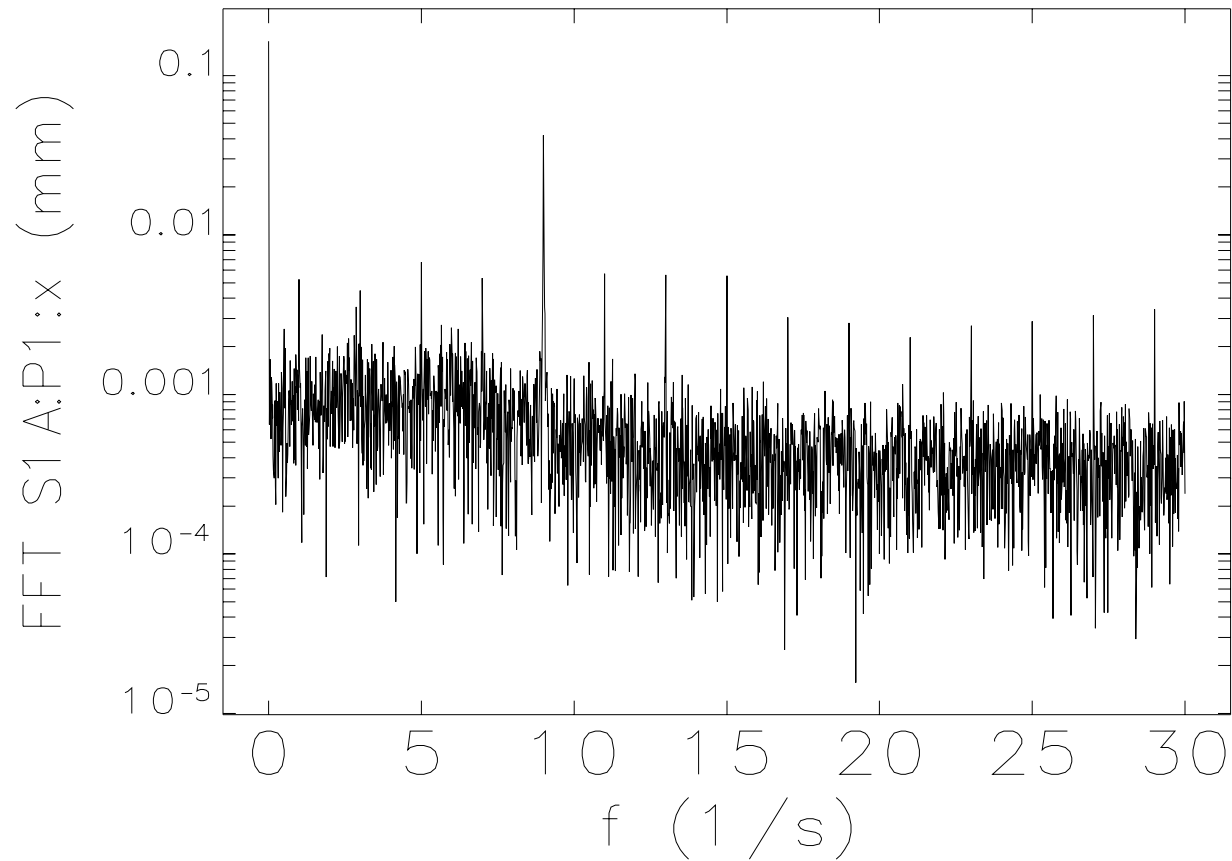
```
sddswmonitor slowAP1.wmon \  
slowAP1.sdds -step=1
```

```
sddsprocess slowAP1.sdds -pipe=out \  
-define=column,t,"Index 60 /",units=s \  
| sddsfft -pipe=in slowAP1.fft \  
-columns=t,*A:P1*
```

```
sddsplot -column=f,FFT*A:P1* slowAP1.fft \  
-separate -mode=y=log,y=specialscales
```

Sar

story



Script for Analysis of Power in Bands

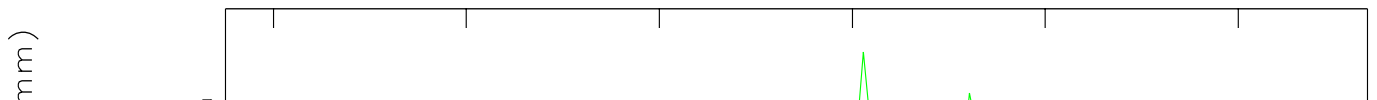
```
#!/bin/sh

#\
exec oagtclsh "$0" "$@"

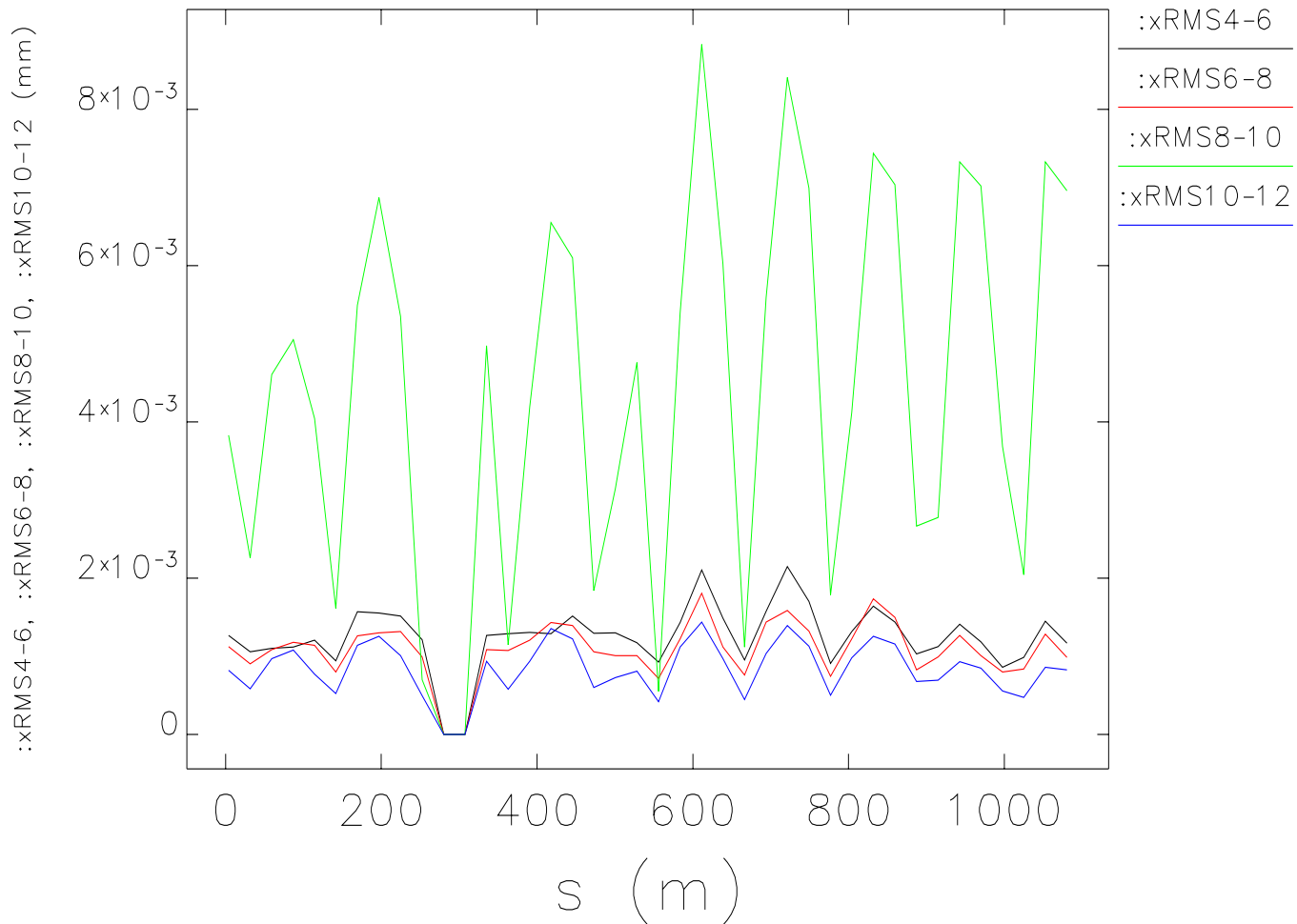
# find RMS amplitude in 2-Hz-wide bands as a function
# of position in the ring

exec sddsprocess slowAP1.fft -pipe=out \
  -process=FFT*,rms,%sRMS4-6,func=f,lower=4,upper=6 \
  -process=FFT*,rms,%sRMS6-8,func=f,lower=6,upper=8 \
  -proc=FFT*,rms,%sRMS8-10,func=f,lower=8,upper=10 \
  -proc=FFT*,rms,%sRMS10-12,func=f,lower=10,upper=12 \
  | sddscollapse -pipe \
  | sddscollect -pipe \
  -collect=suffix=:xRMS4-6 -collect=suffix=:yRMS4-6 \
  -collect=suffix=:xRMS6-8 -collect=suffix=:yRMS6-8 \
  -collect=suffix=:xRMS8-10 -collect=suffix=:yRMS8-10 \
  -collect=suffix=:xRMS10-12 -collect=suffix=:yRMS10-12 \
  | sddsprocess -pipe \
  -edit=column,BPMName,Rootname,%/FFT// \
  | sddsxref -pipe aps.twi -match=BPMName=ElementName \
  -take=s -reuse=rows \
  | sddssort -pipe=in -column=s slowAP1.bands

exec sdds
  -colu
```



Plot of Band Analysis of Beam History



SR Dispersion and Chromaticity

- This experiment involves variation of the rf frequency while measuring tune and orbit.
- **sddsexperiment** is used to drive the experiment and collect orbit data. Two script calls are used by **sddsexperiment** to start the network analyzer sweep and collect tune spectra.
- After completing the experiment, the orbit data is processed using **sddsvslopes** to fit the position vs rf frequency for each BPM in each plane (720 fits). These are then converted to dispersion using the momentum compaction factor (**sddsprocess**).
- The chromaticity analysis script combines the spectra (**sddscombine**), finds the tunes (**sddsprocess**), computes the effective energy offset (**sddsprocess**), and does fits for the chromaticities (**sddspfit**).

PAR Orbit Response

- **sddsexperiment** used to measured response of all BPMs to each corrector.
- To execute the experiment, one uses the command like

sddsexperiment P1H1.exp P1H1.sdds

- A simple script with a loop is used to create each experiment file from a template, then run it.
- To plot the results, use **sddsplot**, e.g.:

sddsplot -column=P1H1,P1P2.x P1H1.sdds

- Data analysis consists of running **sddsslopes** for each corrector pair to find the slope and its error for each BPM.
- The data from a group of scans can be combined for use with **sddscontrollaw** for orbit correction.

Sample sddsexperiment Input File

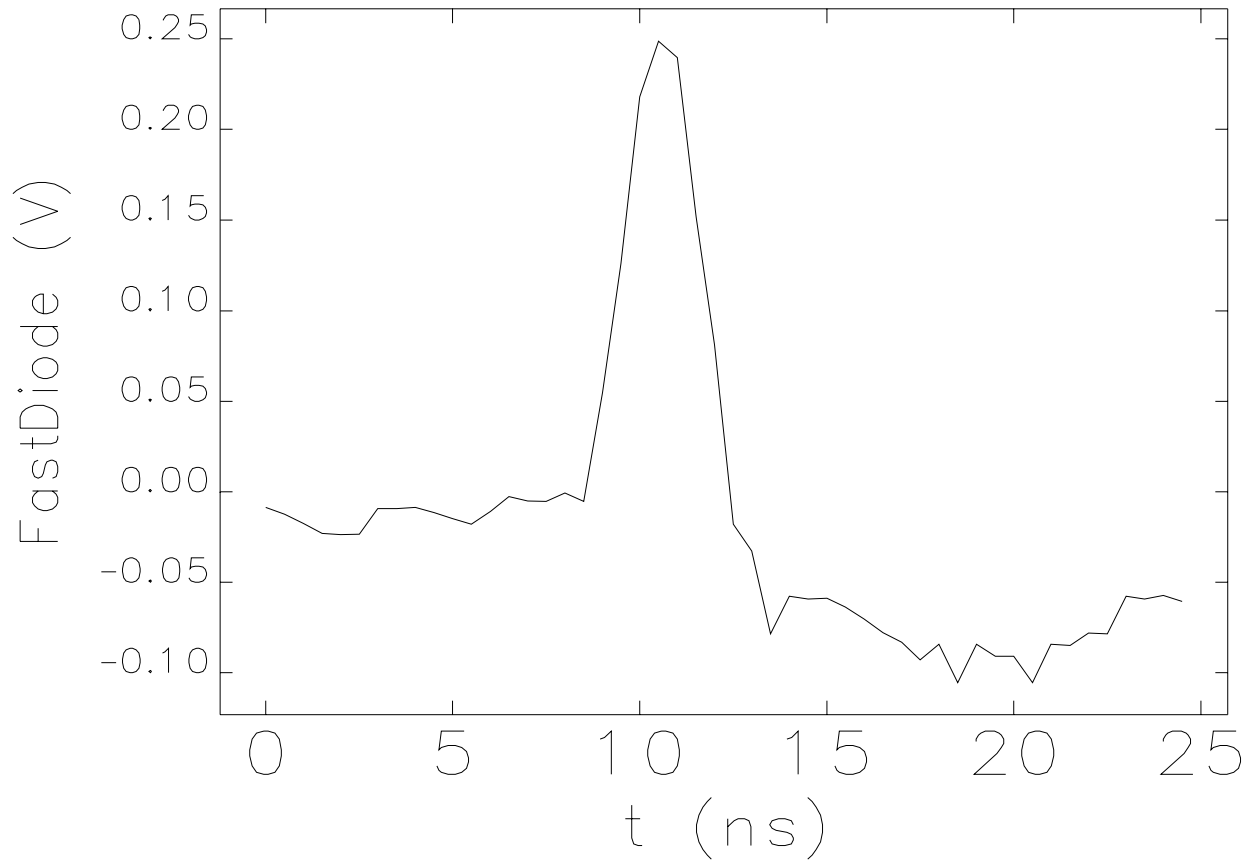
```
! File P1H1.exp
! define what to measure (SDDS file)
&measurement_file
  filename=PBPMs.mon,
  number_to_average=5, include_sigma=1
&end
! define what to vary
&variable
  control_name=P1H1:CurrentAO
  column_name=P1H1, units=A,
  relative_to_original=1,
  index_number=0, index_limit=5,
  initial_value=-2.0, final_value=2.0
&end
! set global parameters and run
&execute
  post_change_pause=3
  intermeasurement_pause=0.5
&end
```

PAR Damping Time

- A fast photodiode viewing synchrotron radiation was digitized with an HP54542A scope using sequential single-shot mode.
- Transferred to workstation via floppy.
- Converted to SDDS with **hpif2sdds**, one segment per page.
- 750 traces viewed with **sddsplot** using X-windows movie feature.
- Processed with **sddsprocess** to invert sign of signal, find baseline and spread, add segment times, etc.
- Analyses plotted with **sddsplot**.
- Each trace fit using **sddsgfit**.
- Fit results collated with **sddscollapse**.
- Processed with **sddsprocess** to put in resolution correction and remove bad points.
- Bunch length and centroid vs time displayed with **sddsplot**.

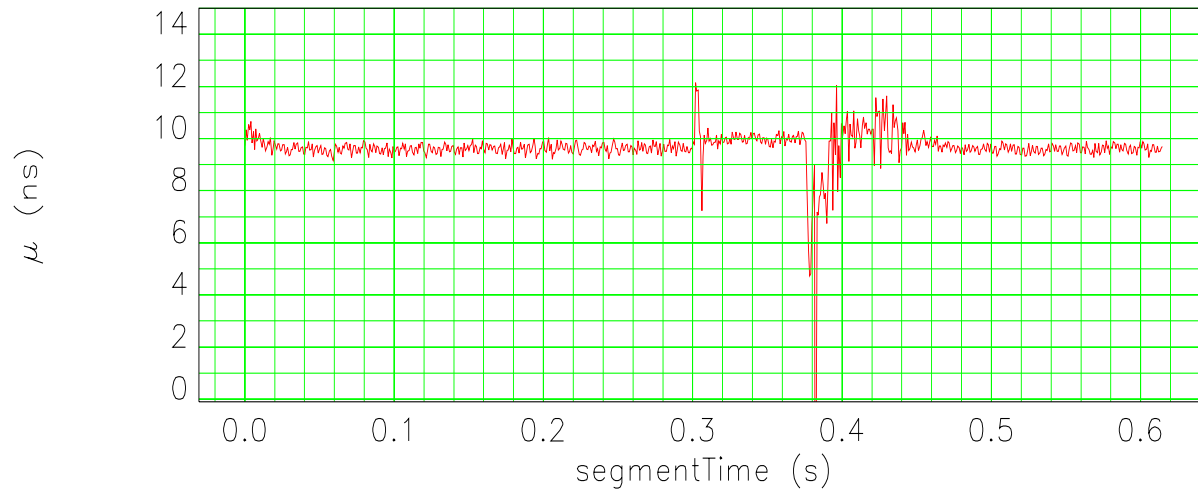
- Part of bunch length data vs time fit with **sddsexpfit** to find longitudinal damping time.
- None of the tools used were developed with this application in mind.
- First analysis took about 15 minutes.

Typical Photodiode Signal —One of 750 Pulses—

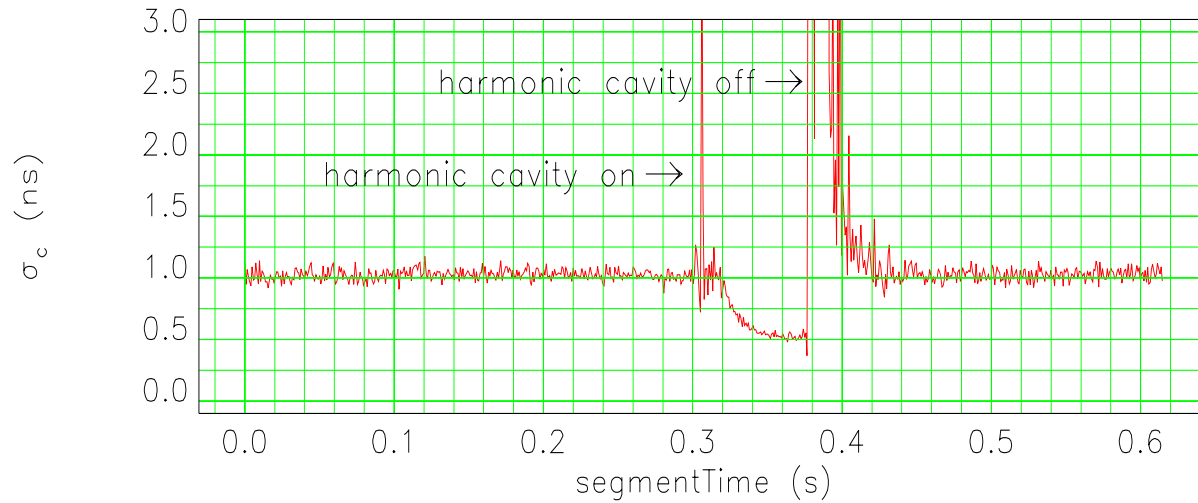


Bunch Parameters vs Time

PAR bunch length and centroid for 4.9 nC

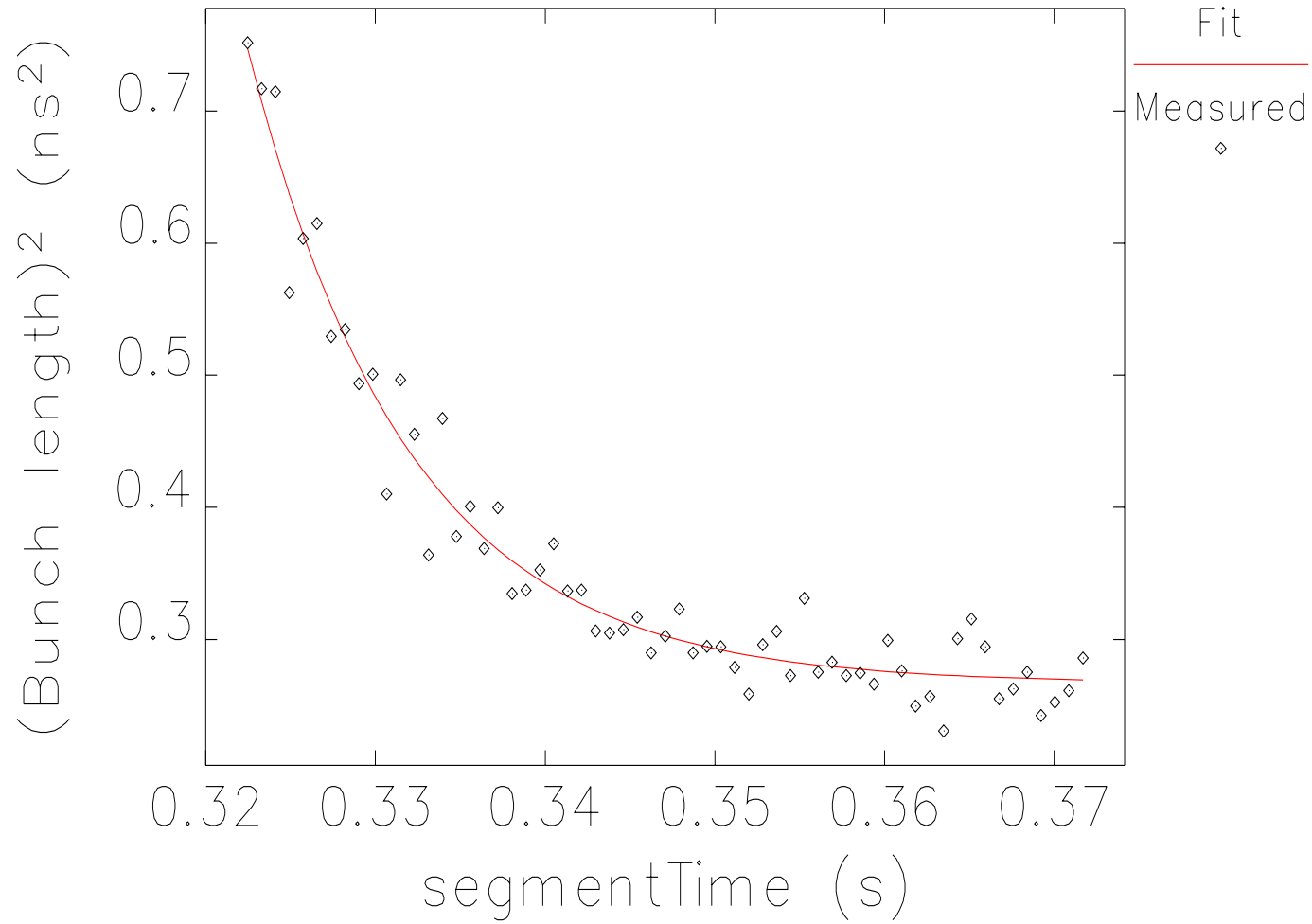


Beam is stored and not extracted



Fit to Obtain Damping Time

Damping time = 18.96 ms



SDDS and Routine Operation

- The work invested in SDDS-based Tcl scripts transfers readily to operations.
- APS depends on SDDS-based Tcl/Tk scripts for routine operation.
- Presently ~26GB of SDDS data for APS operations
 - 48k PVs in configuration management
 - 22k PVs in time-series logging
 - 10k PVs in alarm logging
 - 2k PVs in glitch logging

SDDS-Based Applications Used Routinely by Operations

- Save/Compare/Restore system for managing accelerator configurations.
- Data review and analysis tools for time-series, glitch, alarm, and beam dump data.
- Orbit correction configuration.
- BPM intensity dependence compensation.
- Workstation-based feedback, e.g., orbit correction, rf voltage control, linac energy/trajectory control.
- Steering of ID and BM beamlines.
- Automated startup/setup of power supplies.
- Machine performance analysis and archiving, e.g., RMS beam motion, tunes, fill pattern.
- Workstation screen configuration.