FERMILAB-XXX-AD-APC

Measurement of transverse beam diffusion rates in the Tevatron with collimator scans

G. Stancari,* G. Annala, T. R. Johnson, D. A. Still, and A. Valishev

Fermi National Accelerator Laboratory, P.O. Box 500, Batavia, Illinois 60510, USA

(Dated: April 18, 2012)

The transverse beam diffusion rate vs. particle oscillation amplitude was measured in the Tevatron using collimators. All jaws except one were retracted at the end of regular stores. As the jaw of interest was moved in small steps, the local shower rates were recorded as a function of time. By using a diffusion model, the time evolution of losses could be related to the diffusion rate at the collimator position. The results of these measurements are presented under various experimental conditions.

* e-mail: stancari@fnal.gov

CONTENTS

I. Introduction 3

II. Model 3

III. Methods and data sets 6

IV. Data Analysis 11

V. Results 11

A. Sanity checks 11

B. Physics 15

VI. Discussion and conclusions 22

A. Diagnostic plots 23

1. Data set 8733.5s.T1 23

B. Files and scripts 45

1. dataset_descriptions.txt 45

2. utilities.R 47

3. diana.R 52

References 109

I. INTRODUCTION

Phenomena related to the stochastic transverse beam dynamics in circular accelerators can be described in terms of particle diffusion [1–5]. It was demonstrated that these effects can be observed with collimator scans [11]. Usually, collimator jaws are the devices that are closest to the beam and they define the machine aperture. If they are moved towards the beam center in small steps, typical spikes in the local shower rate are observed, which approach a new steady-state level with a characteristic relaxation time. When collimators are retracted, on the other hand, a dip in losses is observed, which also tends to a new equilibrium level (Figure 3). A detailed description of the Tevatron collimation system can be found in Ref. [6].

These phenomena have been used to estimate the diffusion rate in the beam halo in the SPS at CERN [7], in HERA at DESY [11], and in RHIC at BNL [8]. Similar measurements were carried out at the Tevatron in 2011. Besides the interest in characterizing the beam dynamics of colliding beams, these measurements were motivated by the study of the effects of the novel hollow electron beam collimator [9].

This report describes the measurement technique and the data analysis in detail. Numerical calculations, data analysis, and graphics were done with the open-source, multi-platform statistical package R version 2.14.0 (2011-10-31) [12]. This documentation was produced by integrating LaTeX with R using the Sweave package [13]. The report's source code can be found in the Fermilab Redmine repository: diana.tar.gz. Commented R code chunks can be found throughout this report. The complete analysis scripts are collected in Appendix B.

II. MODEL

A diffusion model of the time evolution of loss rates caused by a step in collimator position was developed [10]. It builds upon the model of Ref. [11] and its assumptions: (1) constant diffusion rate within the range of the step and (2) linear halo tails. These hypotheses allow one to obtain analytical expressions for the solutions of the diffusion equation and for the corresponding loss rates vs. time. The present model addresses some of the limitiations of the previous model and expands it in the following ways: (a) losses before, during, and after the step are predicted; (b) different steady-state rates before and after are explained; (c) determination of the model parameters (diffusion coefficient, tail population, detector calibration, and background rate) is more robust and precise. These calculations are the basis for the measurement of transverse beam diffusion rates as a function

of particle amplitude with collimator scans.

Following Ref. [11], we consider the evolution in time t of a beam of particles with phase-space density $f(J,t)$ described by the diffusion equation

$$\partial_t f = \partial_J \left(D \, \partial_J f \right),$$

where J is the Hamiltonian action and D the diffusion coefficient. The particle flux at a given location $J = J'$ is $\phi = -D \cdot [\partial_J f]_{J=J'}$. During a collimator step, the action $J_c = x_c^2/\beta_c$, corresponding to the collimator position x_c at a ring location where the amplitude function is β_c, changes from its initial value J_{ci} to its final value J_{cf} during a time Δt. The step in action is $\Delta J \equiv J_{cf} - J_{ci}$. In the Tevatron, typical steps are 50 μm in 0.2 s, and the amplitude function is tens of meters. It is assumed that the collimator steps are small enough so that the diffusion coefficient can be treated as a constant in that region. This hypothesis is justified by the fact that the fractional change in action is of the order of $\Delta J_c/J_c \sim (2)(25\ \mu\text{m})/(2\ \text{mm}) = 2.5\%$. Because the diffusion coefficient is a strong function of action ($D \sim J^4$), this translates into a variation of 10% in the diffusion rate, an acceptable systematic in a quantity that varies by orders of magnitude. If D is constant, the diffusion equation becomes $\partial_t f = D \, \partial_{JJ} f$. With these definitions, the particle loss rate at the collimator is equal to the flux at that location: $L = -D \cdot [\partial_J f]_{J=Jc}$. Particle showers caused by the loss of beam are measured with scintillator counters placed close to the collimator jaw. The observed shower rate is parameterized as $S = kL + B$, where k is a normalization constant including detector acceptance and efficiency and B is a background term which includes, for instance, the effect of residual activation. Both k and B are assumed to be independent of collimator position and time during the scan.

Under the hypotheses described above, the diffusion equation can be solved analytically using the method of Green's functions, subject to the boundary condition of vanishing density at the collimator and beyond. Details are given in Ref. [10].

An example of the evolution of the phase-space density according to this model is shown in Figure 1. A few representative snapshots in time are chosen: during collimator movement ($0 \leq t \leq \Delta t$); a short time after the step, with a time scale determined by $\left|J_{ci} - J_{cf}\right|^2/D = 10$ s; and a long time after the step, with a characteristic time $J_c^2/D = 160$ s.

Local losses are proportional to the gradient of the distribution function at the collimator. The gradients differ in the two cases of inward and outward step, denoted by the I and O subscripts, respectively:

$$\partial_J f_I(J_c,t) = -A_i + 2(A_i - A_c) P\left(\frac{-J_c}{\sigma}\right) + \frac{1}{\sqrt{2\pi}\sigma} \cdot$$

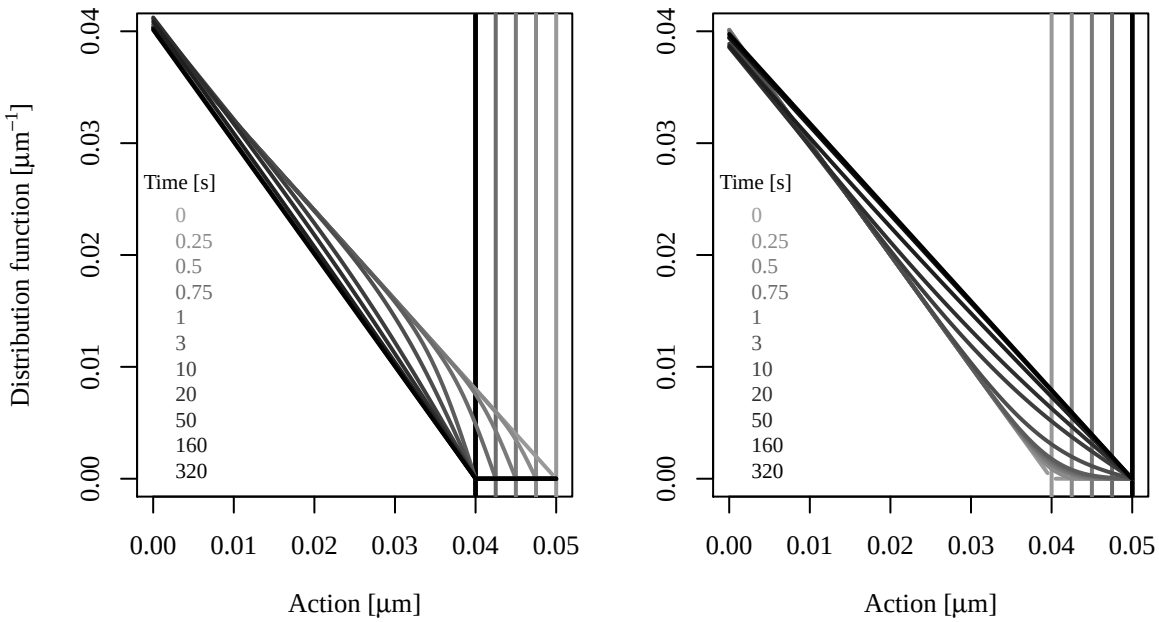


FIG. 1. Evolution of distribution function during collimator step: $f_I(J,t)$ (inward, left) and $f_O(J,t)$ (outward, right). The vertical lines represent the positions of the collimator vs. time. Collimator action varies between $J_{ci} = 0.05$ μm and $J_{cf} = 0.04$ μm in the inward case (viceversa in the outward case) in a time $\Delta t = 1$ s. The slopes of the tails are $A_i = 0.8$ μm^{-2} and $A_f = 1$ μm^{-2} in the inward case (viceversa outwards). The diffusion coefficient is $D = 10^{-5}$ μm^2/s.

$$\cdot \left\{ -2A_i(J_{ci} - J_c) + 2(A_i J_{ci} - A_c J_c) \exp\left[-\frac{1}{2}\left(\frac{J_c}{\sigma}\right)^2 \right] \right\}$$

$$\partial_J f_O(J_c,t) = -2A_i P\left(\frac{J_{ci} - J_c}{\sigma}\right) + 2(A_i - A_c) P\left(\frac{-J_c}{\sigma}\right) + \\ + 2\frac{A_i J_{ci} - A_c J_c}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{J_c}{\sigma}\right)^2 \right].$$

The parameters A_i and A_f are the slopes of the distribution function before and after the step, whereas A_c varies linearly between A_i and A_f as the collimator moves. The parameter σ is defined as $\sigma \equiv \sqrt{2Dt}$; its effect is to expose the dependence of losses on the inverse square root of time, as is typical for diffusion processes. The function $P(x)$ is the S-shaped cumulative Gaussian distribution function: $P(-\infty) = 0$, $P(0) = 1/2$, and $P(\infty) = 1$.

The above expressions are used to model the measured shower rates. A few examples are plotted in Figure 2.

Parameters are estimated from a least-squares fit to the experimental data. The background B is measured before and after the scan when the jaws are retracted. The calibration of kA_i and kA_f

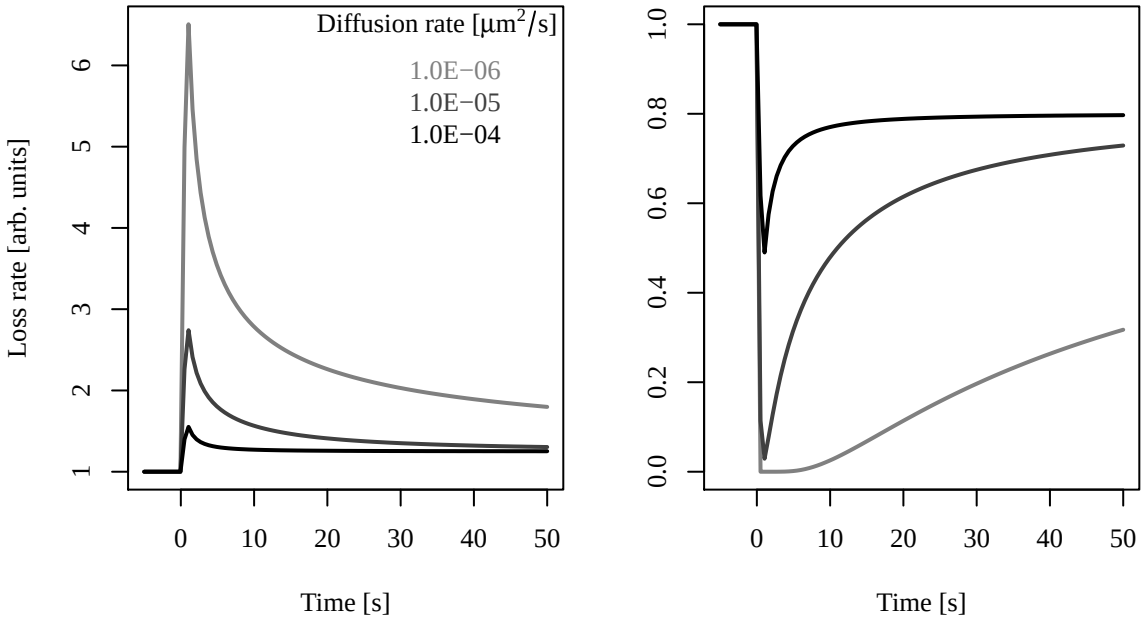


FIG. 2. Calculated evolution of loss rates $L(t)$ during a collimator step according to Eqs. **??**, 1 and 1: inward (left) and outward (right). Collimator action varies between $J_{ci} = 0.05$ μm and $J_{cf} = 0.04$ μm in the inward case (viceversa in the outward case) in a time $\Delta t = 1$ s (see also Figure 1). The effect of 3 different values of the diffusion coefficient D is shown. The slopes of the tails are scaled so that the initial and final steady-state loss rates are the same in all cases: $A_i = 1/D$, $A_f = A_i J_{ci}/J_{cf}$.

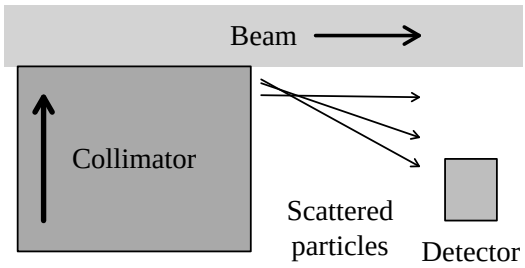
is estimated by comparing the level of losses with beam intensity and lifetime. In practice, D is determined by both the measured relaxation time and by the value of the peak (or dip) in losses.

The model explains the data very well when the diffusion time is long compared to the duration of the step. With this technique, the diffusion rate can be measured over a wide range of amplitudes. At large amplitudes, the method is limited by the vanishing beam population and by the fast diffusion times. The limit at small amplitudes is given by the level of tolerable loss spikes.

III. METHODS AND DATA SETS

Several collimator scans were performed at the Tevatron during 2011. The goal was to observe beam diffusion and to measure the effects of beam-beam forces and of the hollow electron beam scraper. All collimators were retracted except one. The collimator of interest was moved in or out in small steps, and the corresponding local loss rates were recorded as a function of time. The experimental procedure is schematically shown in Figure 3.

Data was collected under different experimental conditions. The individual data sets are de-



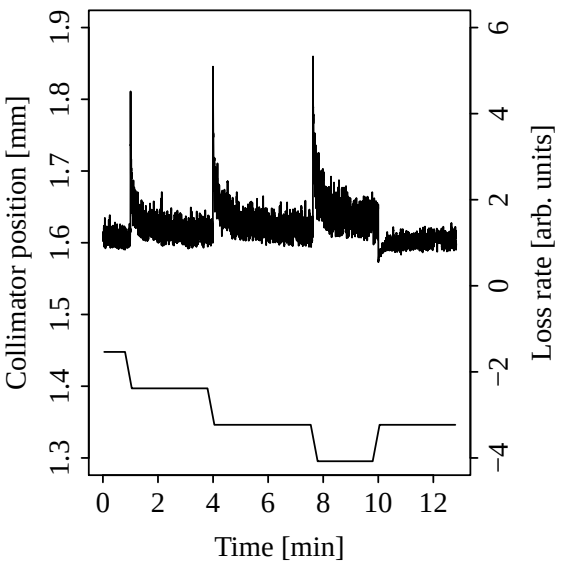


FIG. 3. Schematic diagram of the apparatus (left). Example of the response of local loss rates to inward and outward collimator steps (right).

scribed below and summarized in Tables **??**:

Store 8467, 3 Feb 2011: End of very long store. All collimators out except F49 vertical pbar target. Pulsing on antiproton train 2 (A13–A24) with 3.5σ hole radius (1.29–2.19 mm at F49), 150 mA peak electron current. Vertical beam size at F49 was $\sigma_y = 370$ μm. Vertical collimator steps, 50 μm each in and out to test the technique. Loss monitor at E11 triggered a beam abort during one of the inward steps because protons were being scraped, too (small vertical separation of the orbits at F49).

Store 8508, 17 Feb 2011: Collimator scans in 50-μm steps with F48 vertical pbar absorber: (1) nominal conditions; (2) hollow beam at 3.5σ radius on all 3 trains, 140 mA.

Store 8527, 25 Feb 2011: End of very long store. Collimator scans in 50-μm steps with F48 pbar vertical absorber: (1) nominal conditions; (2) hollow beam at 3.8σ radius on all 3 trains, 100 mA.

Gated loss monitors needed to see effect at full current. Scintillator paddles installed during short March shutdown.

Store 8733, 13 May 2011: Collimator baby steps in and out (F48v, 50 μm) at hole radius of 5σ, acting on pbar bunch train 2 with about 400 mA e-beam current. Collimator baby steps out at hole radius 4σ, train 2, 400 mA.

ID	Store	Date	Time	Collimator	BeamCenter	LossMon	elensI	bunches	Notes
8467.slow	8467	3 Feb 2011	11:20	F49V	-701	S	150	A13–A24	test, aborted
8467.fast	8467	3 Feb 2011	11:20	F49V	-701	T	150	A13–A24	test, aborted
8508.nom	8508	17 Feb 2011	16:00	F48V	382	T	0	A1–A36	nominal
8508.tel	8508	17 Feb 2011	17:20	F48V	382	T	140	A1–A36	TEL2 on, 3.5s
8527.nom	8527	25 Feb 2011	10:35	F48V	377	T	0	A1–A36	nominal
8527.tel	8527	25 Feb 2011	13:00	F48V	377	T	100	A1–A36	TEL2 on, 3.8s
8733.5s.TOT	8733	13 May 2011	11:30	F48V	368	T	0	A1–A12	Total losses
8733.5s.T1	8733	13 May 2011	11:30	F48V	368	G	0	A1–A12	Control train T1
8733.5s.tel	8733	13 May 2011	11:30	F48V	368	G	400	A13–A24	TEL at 5s, 3 TEL trips
8733.5s.T3	8733	13 May 2011	11:30	F48V	368	G	0	A25–A36	Control train T3
8733.4s.TOT	8733	13 May 2011	13:43	F48V	368	T	0	A1–A36	Outward, total losses
8733.4s.T1	8733	13 May 2011	13:43	F48V	368	G	0	A1–A12	Outward, control train T1
8733.4s.tel	8733	13 May 2011	13:43	F48V	368	G	400	A13–A24	TEL at 4s
8733.4s.T3	8733	13 May 2011	13:43	F48V	368	G	0	A25–A36	Outward, control train T3
8749.hi.TOT	8749	20 May 2011	06:57	F48V	369	T	0	A1–A36	total losses
8749.hi.T1	8749	20 May 2011	06:57	F48V	369	G	0	A7–A12	control train T1
8749.hi.tel	8749	20 May 2011	06:57	F48V	369	G	872	A19–A24	1/2 of train 2, 4.25s
8749.hi.T3	8749	20 May 2011	06:57	F48V	369	G	0	A31–A36	control train T3
8749.me.TOT	8749	20 May 2011	09:05	F48V	369	T	0	A1–A36	total losses
8749.me.T1	8749	20 May 2011	09:05	F48V	369	G	0	A7–A12	control train T1
8749.me.tel	8749	20 May 2011	09:05	F48V	369	G	611	A19–A24	TEL2 on 1/2 of train 2
8749.me.T3	8749	20 May 2011	09:05	F48V	369	G	0	A31–A36	control train T3
8749.lo.TOT	8749	20 May 2011	10:00	F48V	369	T	0	A1–A36	total losses
8749.lo.T1	8749	20 May 2011	10:00	F48V	369	G	0	A7–A12	control train T1
8749.lo.tel	8749	20 May 2011	10:00	F48V	369	G	318	A19–A24	TEL2 on 1/2 of train 2
8749.lo.T3	8749	20 May 2011	10:00	F48V	369	G	0	A31–A36	control train T3
8764.hin.TOT	8764	24 May 2011	14:53	F48V	369	T	0	A1–A4	pbar only, h co. in, total
8764.hin.nom	8764	24 May 2011	14:53	F48V	369	G	0	A1–A4	pbar only, h co. in, control
8764.hin.tel	8764	24 May 2011	14:53	F48V	369	G	832	A13–A16	pbar only, h co. in, 4.5s
8764.TOT	8764	24 May 2011	17:58	F48V	369	T	0	A1–A4	pbar only, total losses
8764.nom	8764	24 May 2011	17:58	F48V	369	G	0	A1–A4	pbar only, control bunches
8764.tel	8764	24 May 2011	17:58	F48V	369	G	832	A13–A16	pbar only, 4.5s

TABLE I. Data set summary.

ID	Date	Time	Bg	Bm	eVm	eIm	eIin	rig	rog	ri	ris	ro	ros
8467.slow	3 Feb 2011	11:20	3.9	12.7	1.6	31.1	150	4.5	7.62	2.5	3.53	4.23	5.98
8467.fast	3 Feb 2011	11:20	3.9	12.7	1.6	31.1	150	4.5	7.62	2.5	3.54	4.23	6
8508.nom	17 Feb 2011	16:00	3.81	21.4	3.92	12.7	0	4.5	7.62	1.9	3.4	3.22	5.76
8508.tel	17 Feb 2011	17:20	3.91	20	4.74	86.1	140	4.5	7.62	1.99	3.49	3.37	5.91
8527.nom	25 Feb 2011	10:35	3.91	10	3.6	12.2	0	4.5	7.62	2.81	3.75	4.76	6.35
8527.tel	25 Feb 2011	13:00	3.91	9.98	4.78	75.1	100	4.5	7.62	2.82	3.75	4.77	6.35
8733.5s.TOT	13 May 2011	11:30	3.9	10.2	4.78	82.6	0	4.5	7.62	2.78	4.83	4.71	8.18
8733.5s.T1	13 May 2011	11:30	3.9	10.2	4.78	82.6	0	4.5	7.62	2.78	4.83	4.71	8.18
8733.5s.tel	13 May 2011	11:30	3.9	10.2	4.78	82.6	400	4.5	7.62	2.78	4.88	4.71	8.26
8733.5s.T3	13 May 2011	11:30	3.9	10.2	4.78	82.6	0	4.5	7.62	2.78	4.82	4.71	8.17
8733.4s.TOT	13 May 2011	13:43	3.9	14.7	4.37	83.3	0	4.5	7.62	2.32	3.95	3.93	6.7
8733.4s.T1	13 May 2011	13:43	3.9	14.7	4.37	83.3	0	4.5	7.62	2.32	3.94	3.93	6.66
8733.4s.tel	13 May 2011	13:43	3.9	14.7	4.37	83.3	400	4.5	7.62	2.32	3.99	3.93	6.75
8733.4s.T3	13 May 2011	13:43	3.9	14.7	4.37	83.3	0	4.5	7.62	2.32	3.94	3.93	6.68
8749.hi.TOT	20 May 2011	06:57	3.9	12.4	4.79	74	0	4.5	7.62	2.53	4.07	4.28	6.89
8749.hi.T1	20 May 2011	06:57	3.9	12.4	4.79	74	0	4.5	7.62	2.53	4.08	4.28	6.9
8749.hi.tel	20 May 2011	06:57	3.9	12.4	4.79	74	872	4.5	7.62	2.53	4.07	4.28	6.9
8749.hi.T3	20 May 2011	06:57	3.9	12.4	4.79	74	0	4.5	7.62	2.53	4.12	4.28	6.97
8749.me.TOT	20 May 2011	09:05	3.9	12.4	3.36	51.9	0	4.5	7.62	2.53	4	4.28	6.77
8749.me.T1	20 May 2011	09:05	3.9	12.4	3.36	51.9	0	4.5	7.62	2.53	4	4.28	6.78
8749.me.tel	20 May 2011	09:05	3.9	12.4	3.36	51.9	611	4.5	7.62	2.53	4.01	4.28	6.8
8749.me.T3	20 May 2011	09:05	3.9	12.4	3.36	51.9	0	4.5	7.62	2.53	4.05	4.28	6.85
8749.lo.TOT	20 May 2011	10:00	3.9	12.4	2.11	26.8	0	4.5	7.62	2.53	3.96	4.28	6.7
8749.lo.T1	20 May 2011	10:00	3.9	12.4	2.11	26.8	0	4.5	7.62	2.53	3.96	4.28	6.7
8749.lo.tel	20 May 2011	10:00	3.9	12.4	2.11	26.8	318	4.5	7.62	2.53	3.99	4.28	6.76
8749.lo.T3	20 May 2011	10:00	3.9	12.4	2.11	26.8	0	4.5	7.62	2.53	3.99	4.28	6.76
8764.hin.TOT	24 May 2011	14:53	3.9	13	4.25	52.9	0	4.5	7.62	2.47	4.42	4.18	7.48
8764.hin.nom	24 May 2011	14:53	3.9	13	4.25	52.9	0	4.5	7.62	2.47	4.42	4.18	7.48
8764.hin.tel	24 May 2011	14:53	3.9	13	4.25	52.9	832	4.5	7.62	2.47	4.5	4.18	7.61
8764.TOT	24 May 2011	17:58	3.9	13	4.25	52.9	0	4.5	7.62	2.47	4.37	4.18	7.39
8764.nom	24 May 2011	17:58	3.9	13	4.25	52.9	0	4.5	7.62	2.47	4.37	4.18	7.39
8764.tel	24 May 2011	17:58	3.9	13	4.25	52.9	832	4.5	7.62	2.47	4.45	4.18	7.53

TABLE II. Summary of electron lens conditions.

ID	Date	Time	bunches	Nm	dppm	sxSLm	sxC	sxTEL2	sySLm	syC	syTEL2
8467.slow	3 Feb 2011	11:20	A13–A24	314	1.6	0.427	0.828	0.545	0.589	0.365	0.707
8467.fast	3 Feb 2011	11:20	A13–A24	309	1.6	0.426	0.826	0.544	0.587	0.364	0.704
8508.nom	17 Feb 2011	16:00	A1–A36	649	1.44	0.329	0.448	0.412	0.464	0.238	0.558
8508.tel	17 Feb 2011	17:20	A1–A36	617	1.45	0.334	0.455	0.418	0.474	0.243	0.57
8527.nom	25 Feb 2011	10:35	A1–A36	1160	1.71	0.452	0.641	0.576	0.625	0.321	0.75
8527.tel	25 Feb 2011	13:00	A1–A36	1124	1.71	0.453	0.642	0.578	0.627	0.322	0.752
8733.5s.TOT	13 May 2011	11:30	A1–A12	191	1.29	0.324	0.452	0.41	0.48	0.247	0.576
8733.5s.T1	13 May 2011	11:30	A1–A12	191	1.29	0.324	0.452	0.41	0.48	0.247	0.576
8733.5s.tel	13 May 2011	11:30	A13–A24	201	1.3	0.323	0.451	0.409	0.476	0.245	0.571
8733.5s.T3	13 May 2011	11:30	A25–A36	200	1.32	0.326	0.454	0.413	0.481	0.247	0.577
8733.4s.TOT	13 May 2011	13:43	A1–A36	561	1.32	0.331	0.462	0.419	0.489	0.251	0.586
8733.4s.T1	13 May 2011	13:43	A1–A12	181	1.31	0.33	0.462	0.419	0.491	0.253	0.589
8733.4s.tel	13 May 2011	13:43	A13–A24	189	1.32	0.328	0.458	0.415	0.485	0.249	0.582
8733.4s.T3	13 May 2011	13:43	A25–A36	190	1.34	0.334	0.466	0.423	0.49	0.252	0.588
8749.hi.TOT	20 May 2011	06:57	A1–A36	1915	1.52	0.383	0.536	0.485	0.516	0.265	0.621
8749.hi.T1	20 May 2011	06:57	A7–A12	331	1.53	0.385	0.539	0.488	0.515	0.265	0.62
8749.hi.tel	20 May 2011	06:57	A19–A24	326	1.52	0.383	0.536	0.486	0.516	0.265	0.62
8749.hi.T3	20 May 2011	06:57	A31–A36	323	1.53	0.383	0.535	0.485	0.51	0.262	0.614
8749.me.TOT	20 May 2011	09:05	A1–A36	1837	1.54	0.389	0.545	0.494	0.525	0.27	0.632
8749.me.T1	20 May 2011	09:05	A7–A12	317	1.55	0.393	0.55	0.498	0.525	0.27	0.631
8749.me.tel	20 May 2011	09:05	A19–A24	310	1.54	0.388	0.543	0.492	0.523	0.269	0.629
8749.me.T3	20 May 2011	09:05	A31–A36	310	1.54	0.389	0.544	0.493	0.519	0.267	0.624
8749.lo.TOT	20 May 2011	10:00	A1–A36	1797	1.54	0.392	0.55	0.498	0.531	0.273	0.638
8749.lo.T1	20 May 2011	10:00	A7–A12	311	1.56	0.395	0.554	0.502	0.531	0.273	0.639
8749.lo.tel	20 May 2011	10:00	A19–A24	303	1.55	0.391	0.548	0.496	0.527	0.271	0.633
8749.lo.T3	20 May 2011	10:00	A31–A36	304	1.55	0.392	0.549	0.497	0.526	0.271	0.633
8764.hin.TOT	24 May 2011	14:53	A1–A4	113	1.37	0.359	0.507	0.457	0.465	0.239	0.559
8764.hin.nom	24 May 2011	14:53	A1–A4	113	1.37	0.359	0.507	0.457	0.465	0.239	0.559
8764.hin.tel	24 May 2011	14:53	A13–A16	110	1.35	0.352	0.497	0.448	0.456	0.235	0.549
8764.TOT	24 May 2011	17:58	A1–A4	112	1.4	0.373	0.529	0.476	0.47	0.241	0.565
8764.nom	24 May 2011	17:58	A1–A4	112	1.4	0.373	0.529	0.476	0.47	0.241	0.565
8764.tel	24 May 2011	17:58	A13–A16	109	1.38	0.366	0.519	0.467	0.461	0.237	0.554

TABLE III. Summary of beam conditions.

Store 8749, 20 May 2011: Collimator scans (F48v) with hollow electron beam on bunches A19–A24 of train 2 with 3 different peak currents: 872 mA (inward and outward), 611 mA, and 318 mA (a few steps inward, then outward). The scans spanned about 1.2 mm (more than 4σ).

Store 8764, 24 May 2011 (pbar only): Pbar only store with bunches A1–A4 and A13–A16. Measured removal rates, losses, and emittances with elens on A13–A16, skip 0, 832 mA peak, 4.5σ hole radius (4-13-4 kG). Measured diffusion rates vs. amplitude for affected and control trains with F48 vertical collimator steps. Scan was done twice to estimate effect of horizontal collimator (half way in the first time, out the second time). A couple of fast collimator scans (inward and outward) to estimate halo populations.

	β_x [m]	β_y [m]	D_x [m]	D_y [m]
F48	93.2	29.1	0.757	−0.274
F49	170.7	40.4	1.23	−0.584
TEL2	67.9	153.2	1.25	−1.01
p SL	52.7	114.6	1.67	−0.576
$\bar{p}$ SL	37.6	109.3	1.25	−0.603

TABLE IV. Tevatron lattice functions at the collimators (F48 and F49), the electron lens (TEL2), and (anti)proton synchrotron light monitors (measurements of 20 Aug 2010).

The Tevatron lattice functions used in this analysis are shown in Table IV.

IV. DATA ANALYSIS

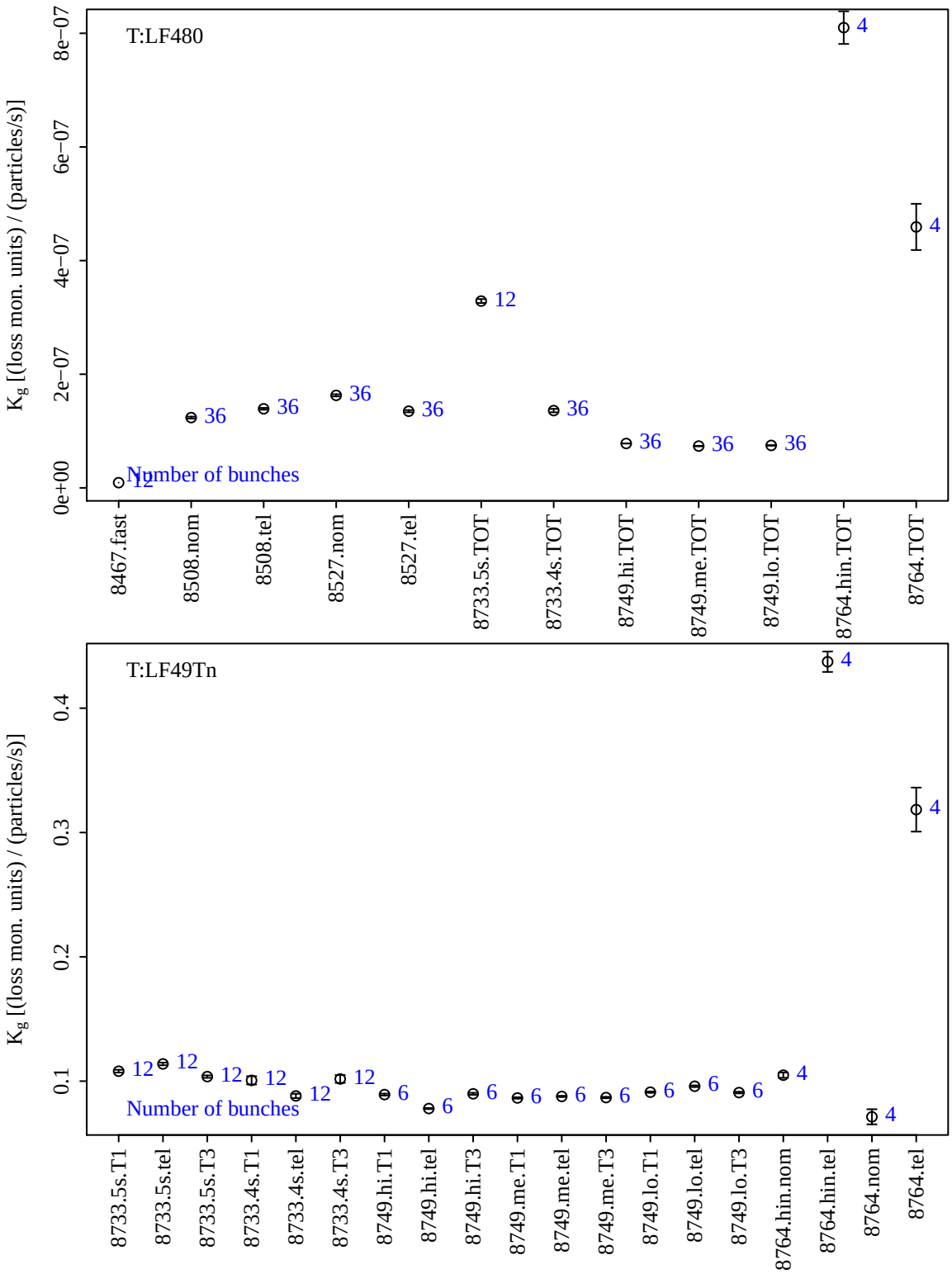
Description of data analysis.

V. RESULTS

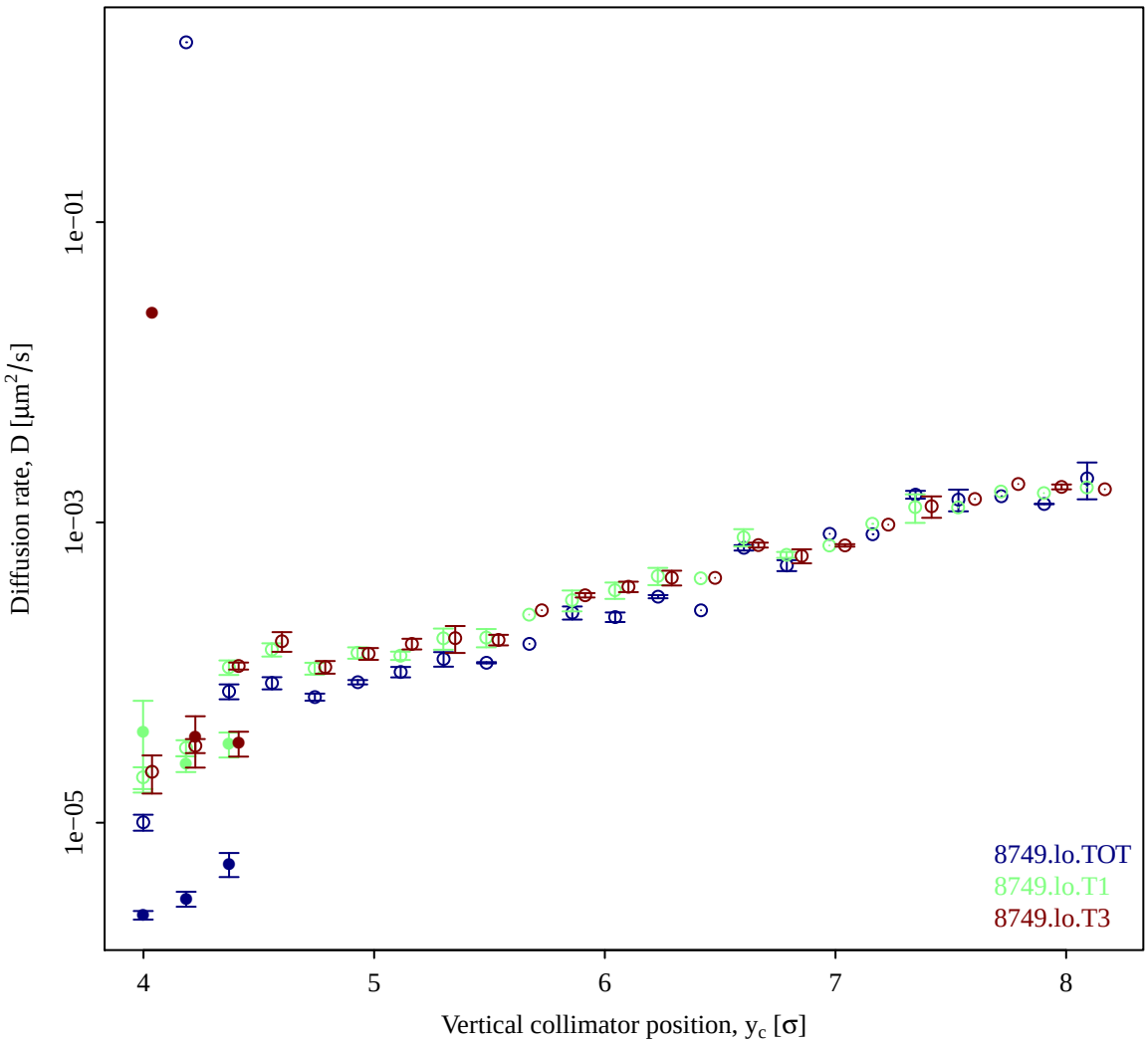
A. Sanity checks

Calibration of losses.

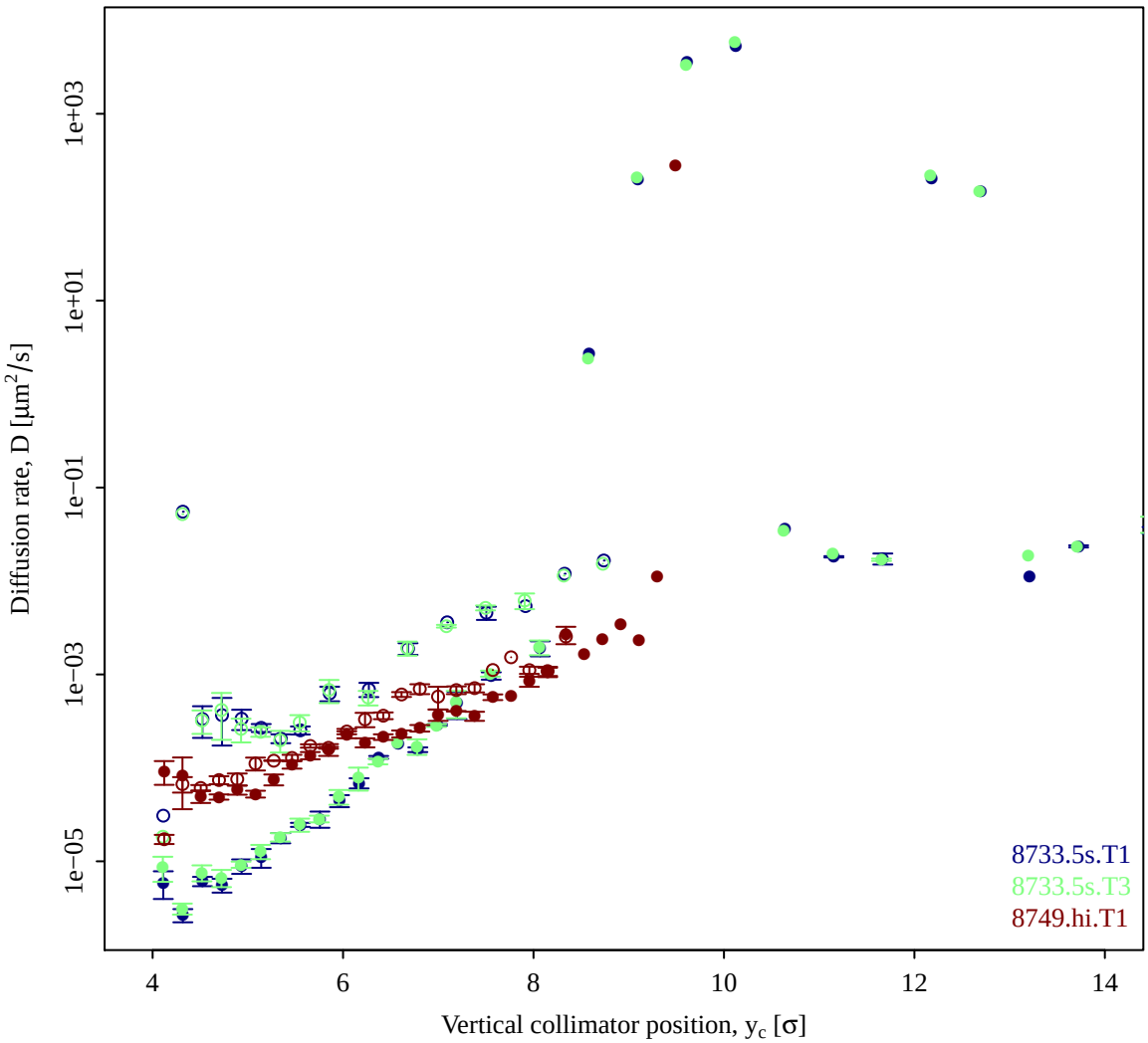
pdf 2 null device 1

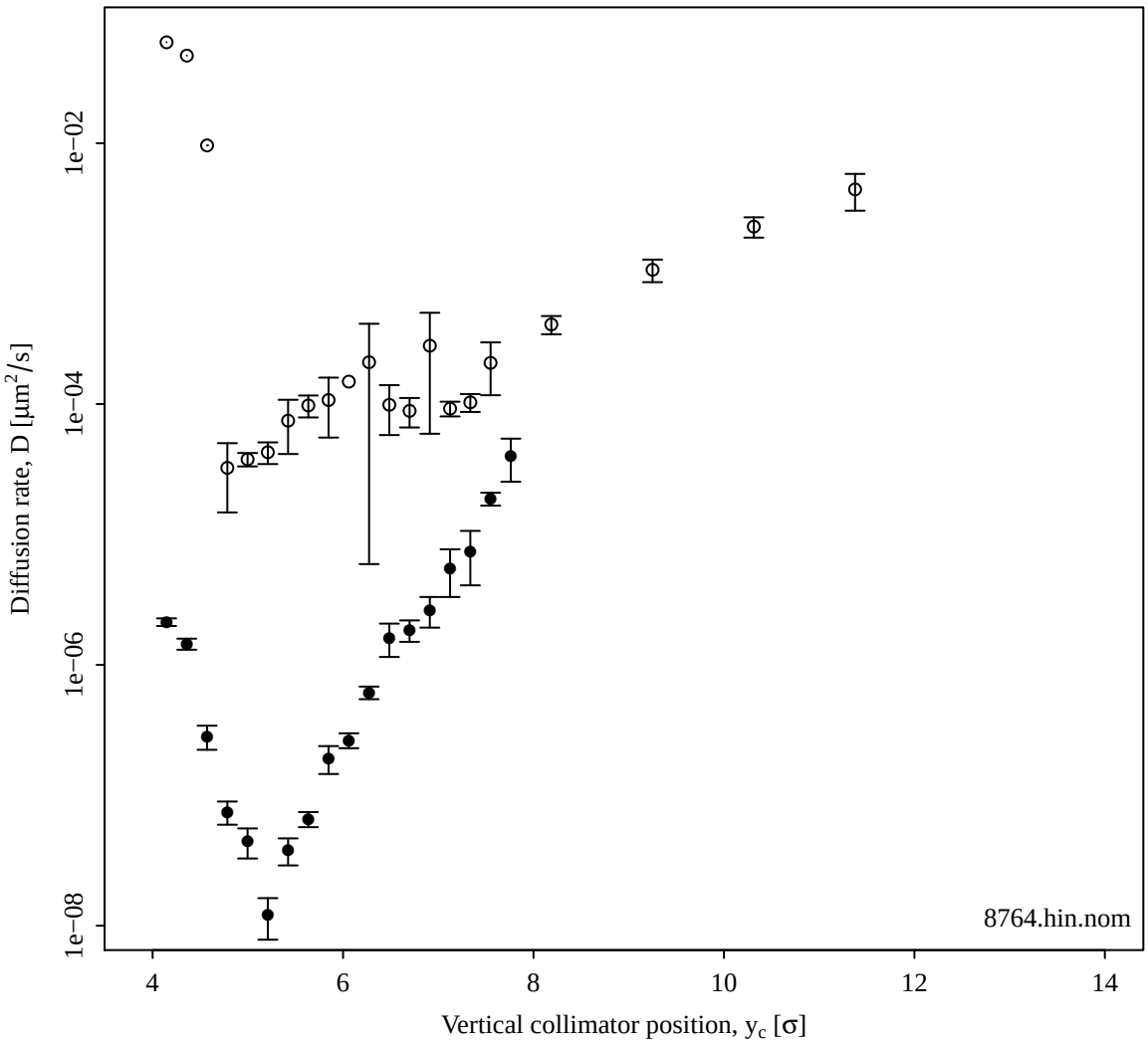


Total vs. gated loss monitors.



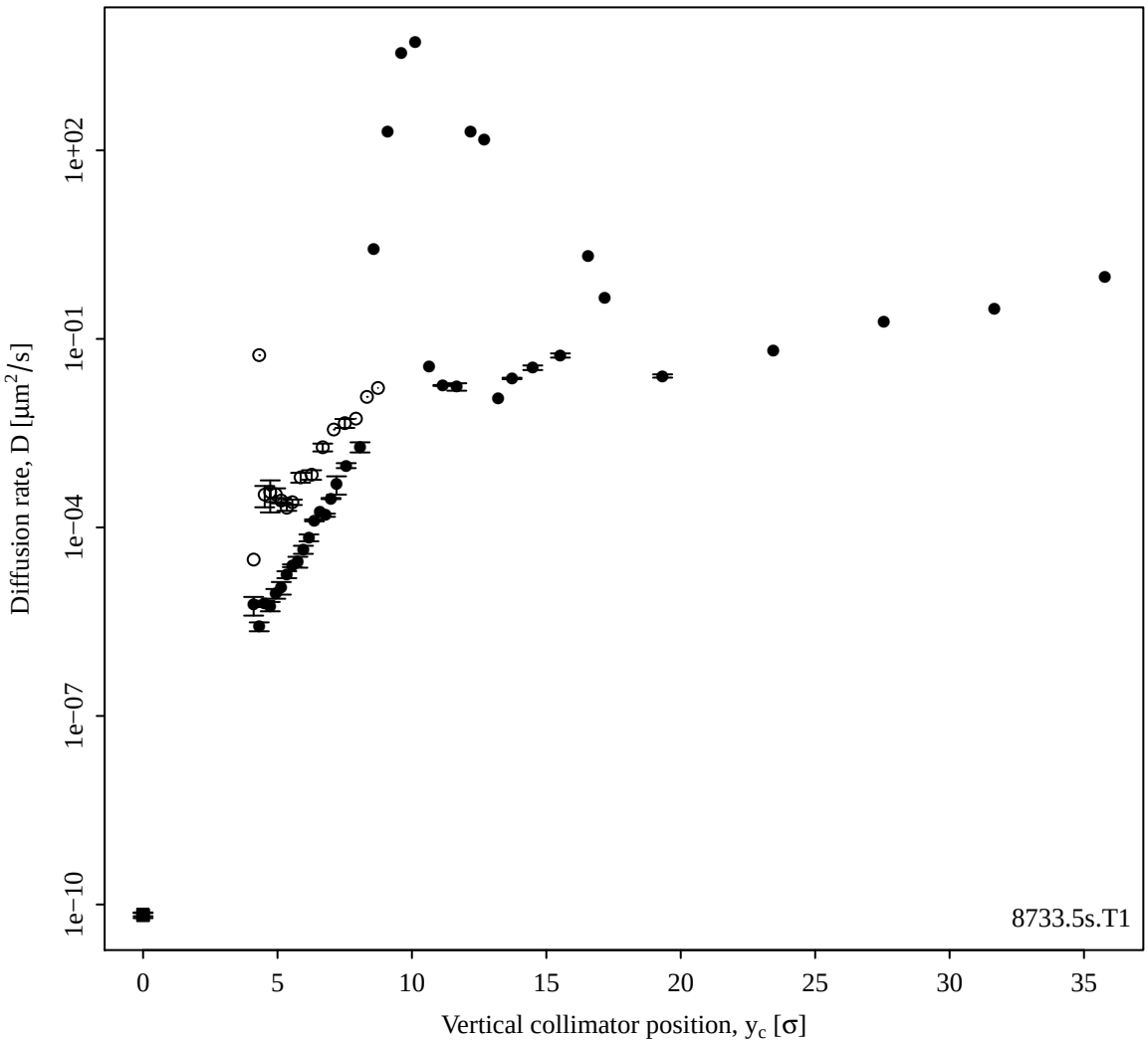
Inward vs. outward.


Diffusion rate, D [μm²/s]
1e+03
1e+01
1e−01
1e−03
1e−05
4
6
8
10
12
14
Vertical collimator position, y_c [σ]
8733.5s.T1
8733.5s.T3
8749.hi.T1

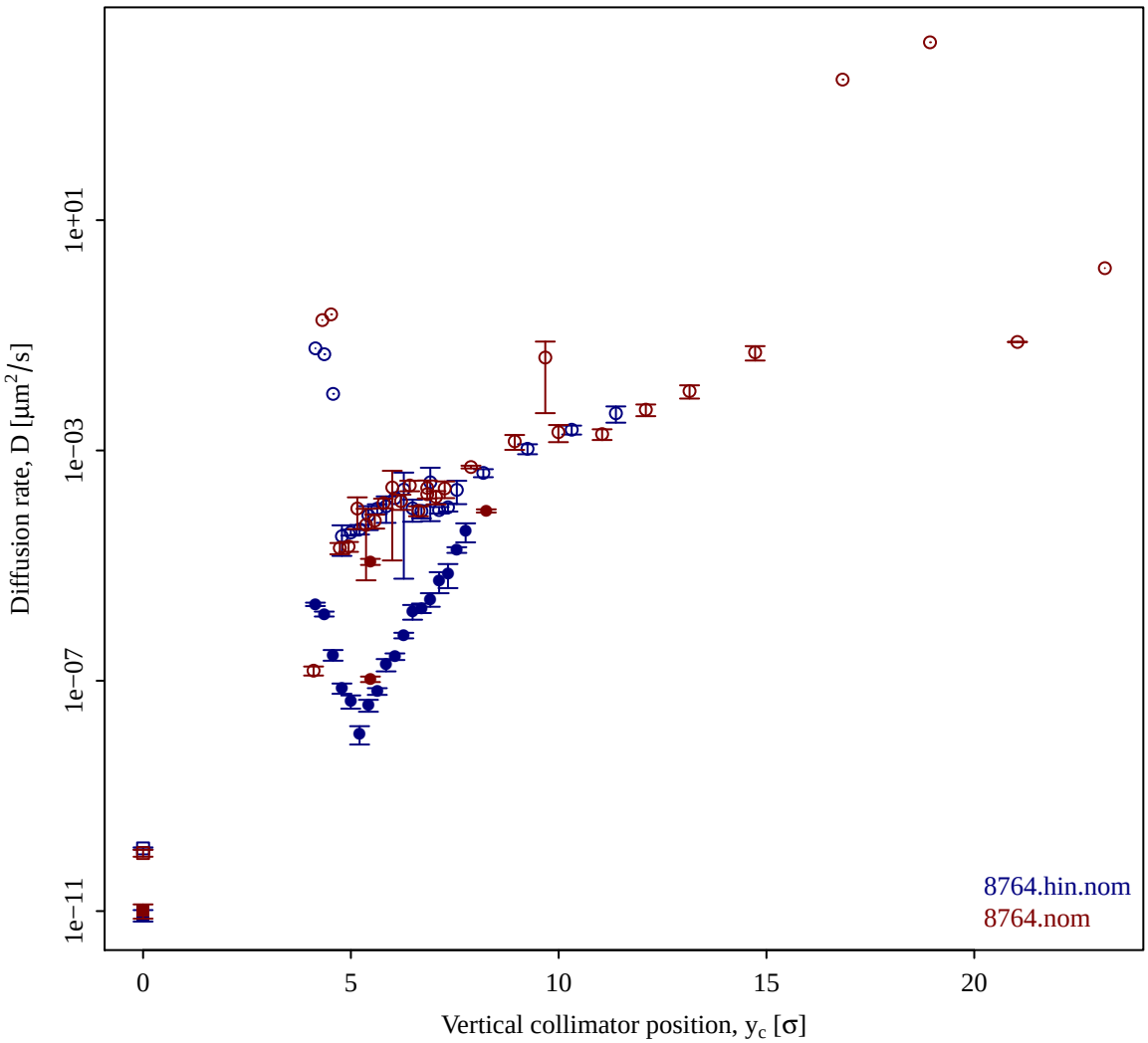


B. Physics

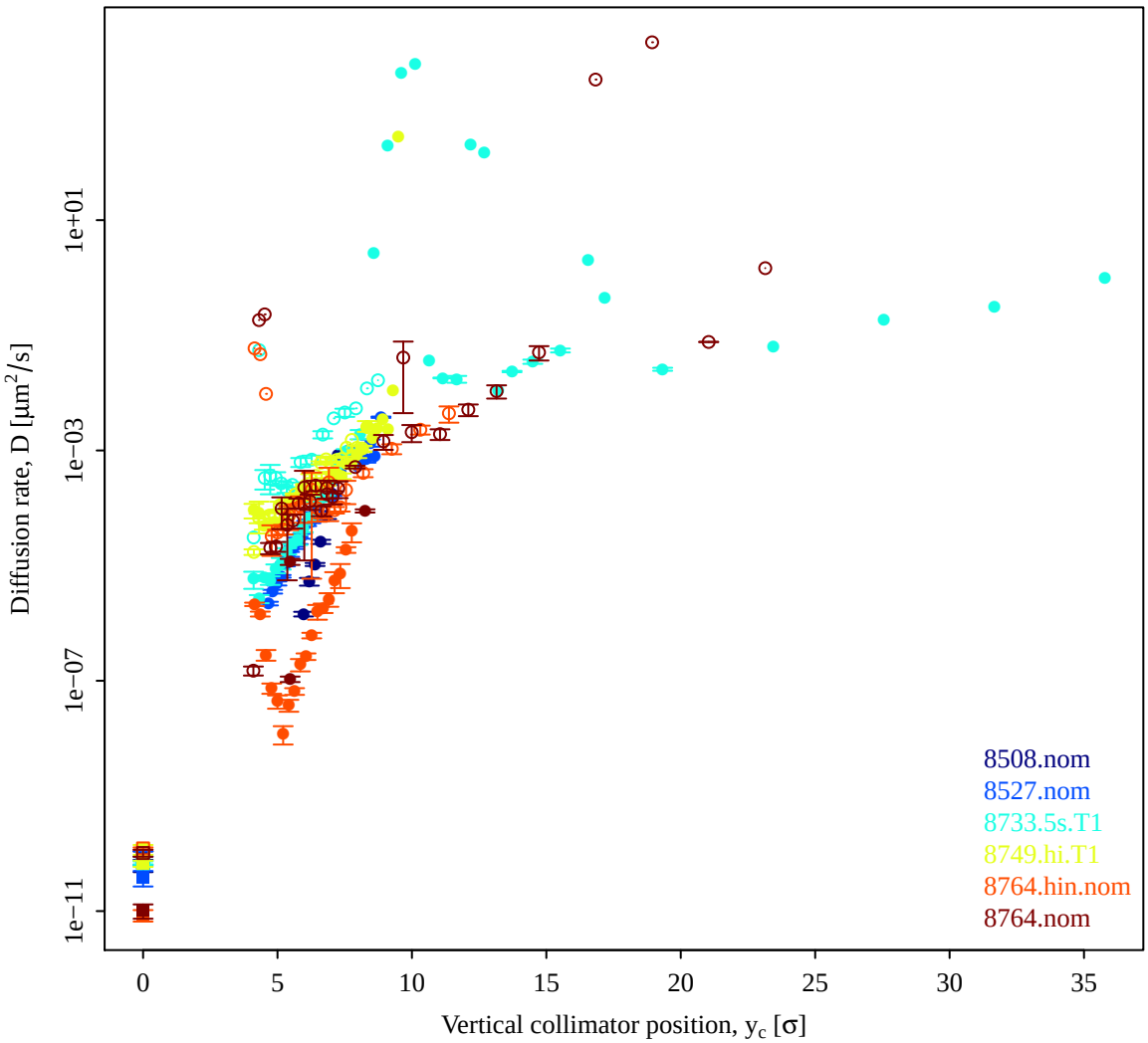
Wide range.

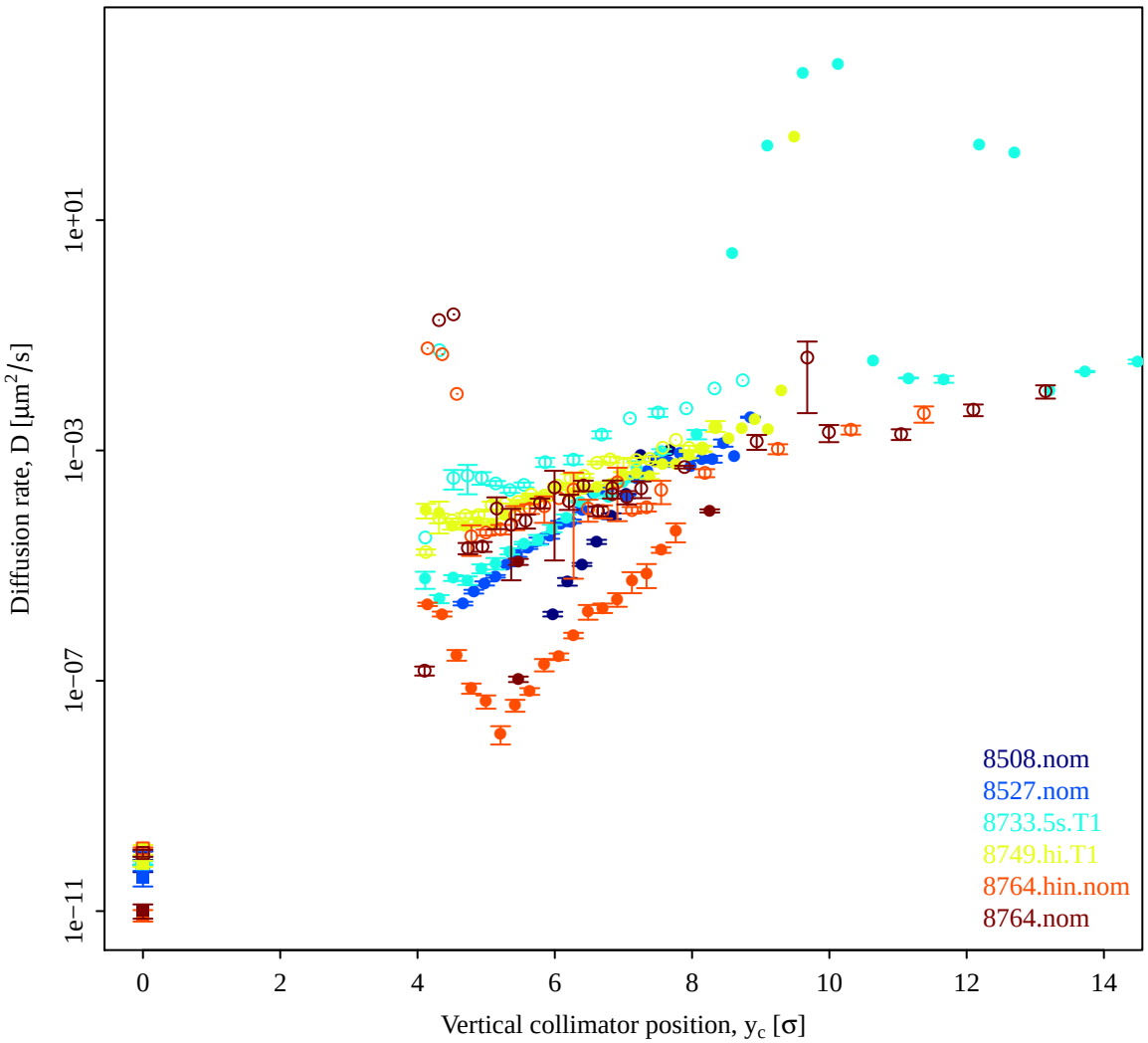

Diffusion rate, D [μm²/s]
1e+02
1e−01
1e−04
1e−07
1e−10
8733.5s.T1
0
5
10
15
20
25
30
35
Vertical collimator position, y_c [σ]

Pbar only.

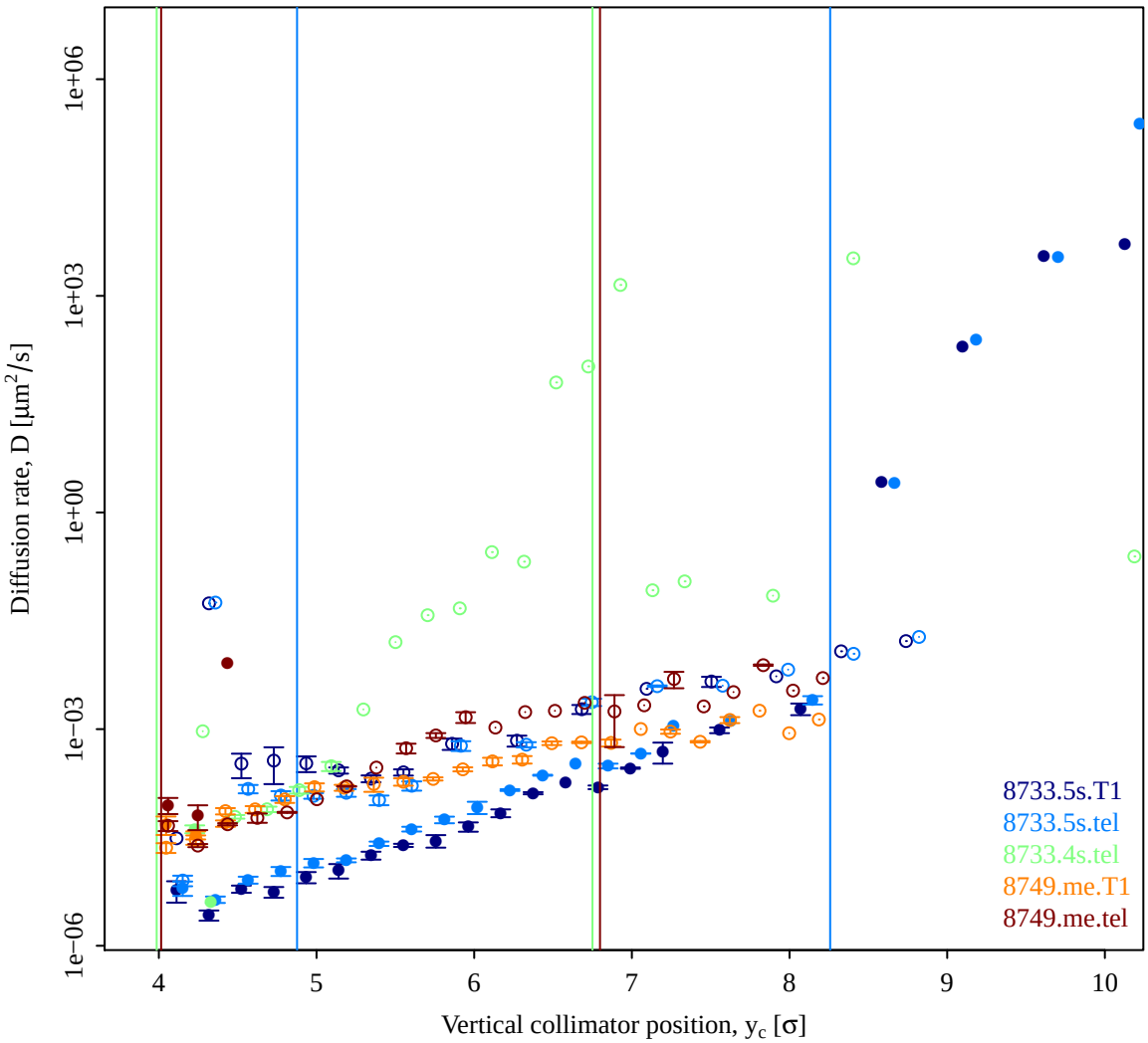


Collisions vs. pbar only.

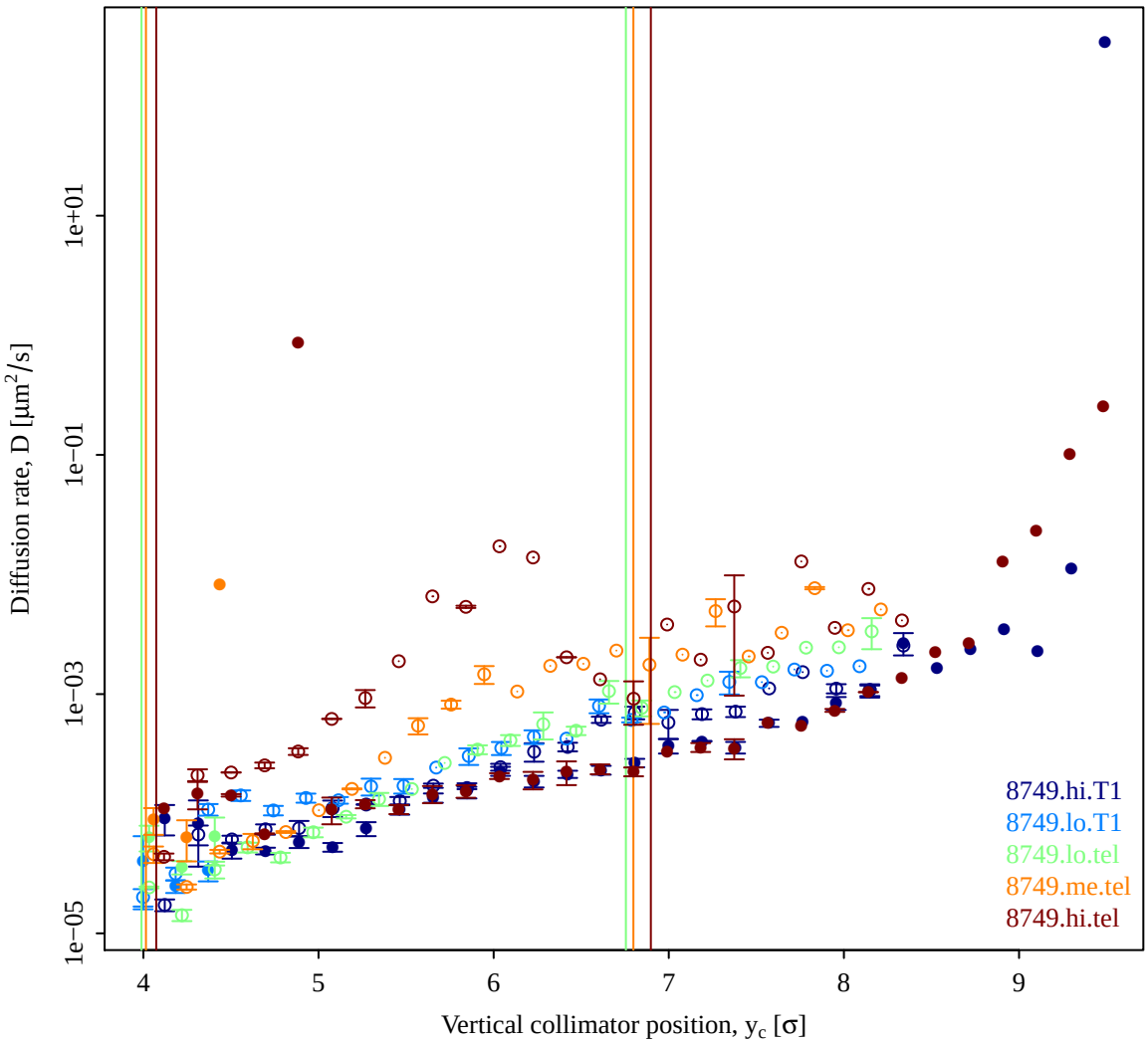

Diffusion rate, D [μm²/s]
1e+01
1e−03
1e−07
1e−11
0
5
10
15
20
25
30
35
Vertical collimator position, y_c [σ]
8508.nom
8527.nom
8733.5s.T1
8749.hi.T1
8764.hin.nom
8764.nom



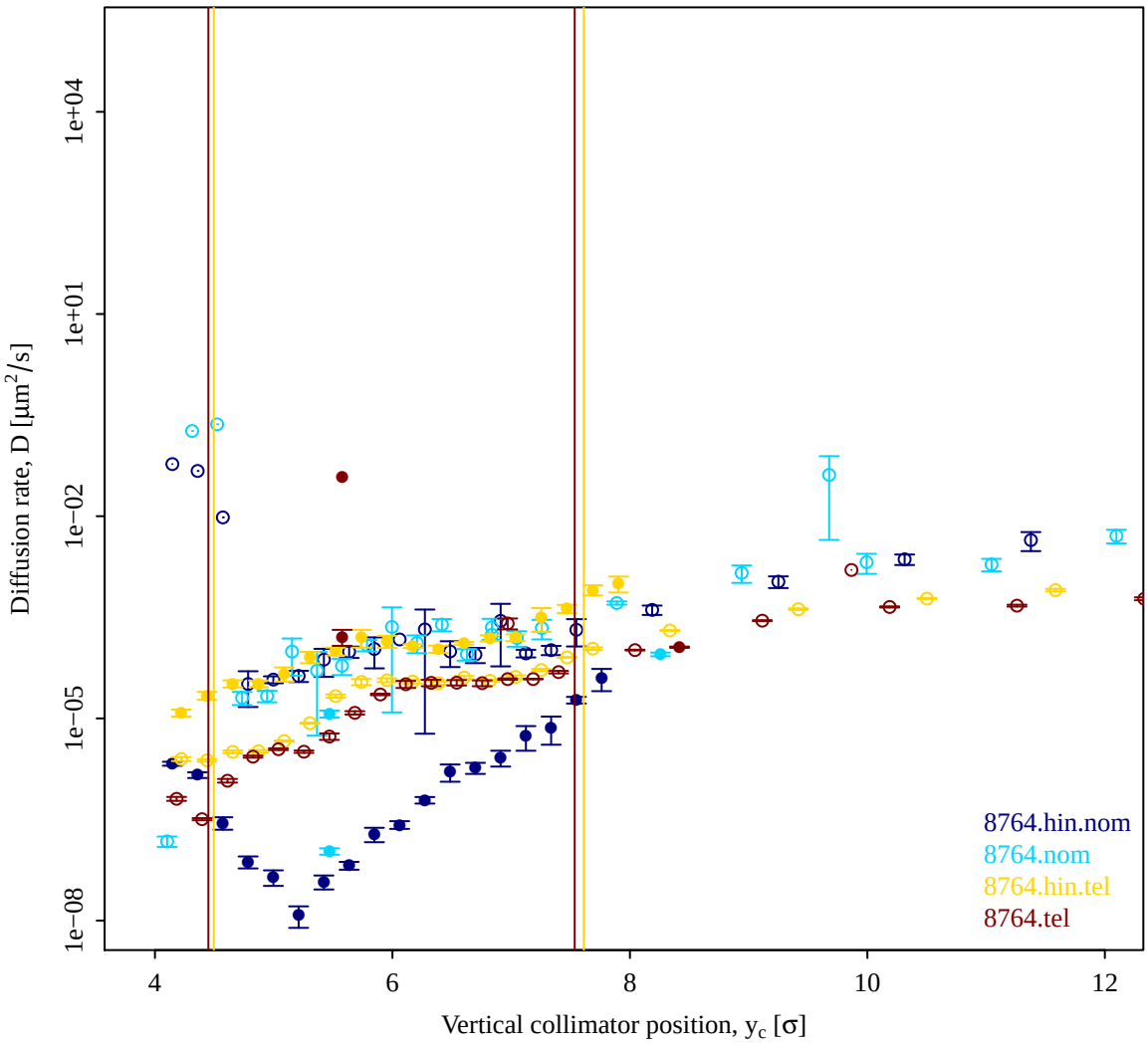
E-lens position.



E-lens intensity.



E-lens in pbar-only.



VI. DISCUSSION AND CONCLUSIONS

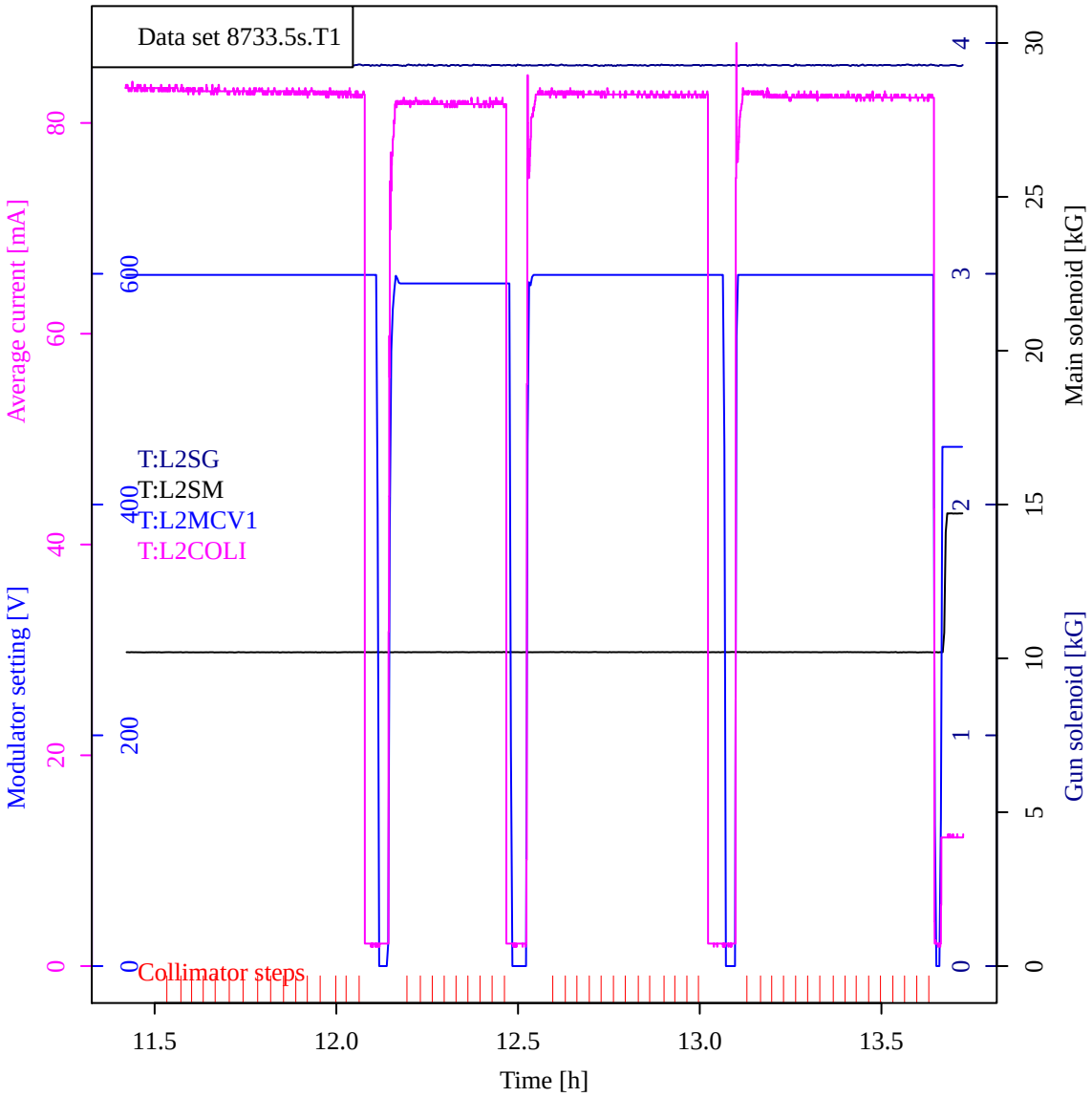
Why are in and out different?

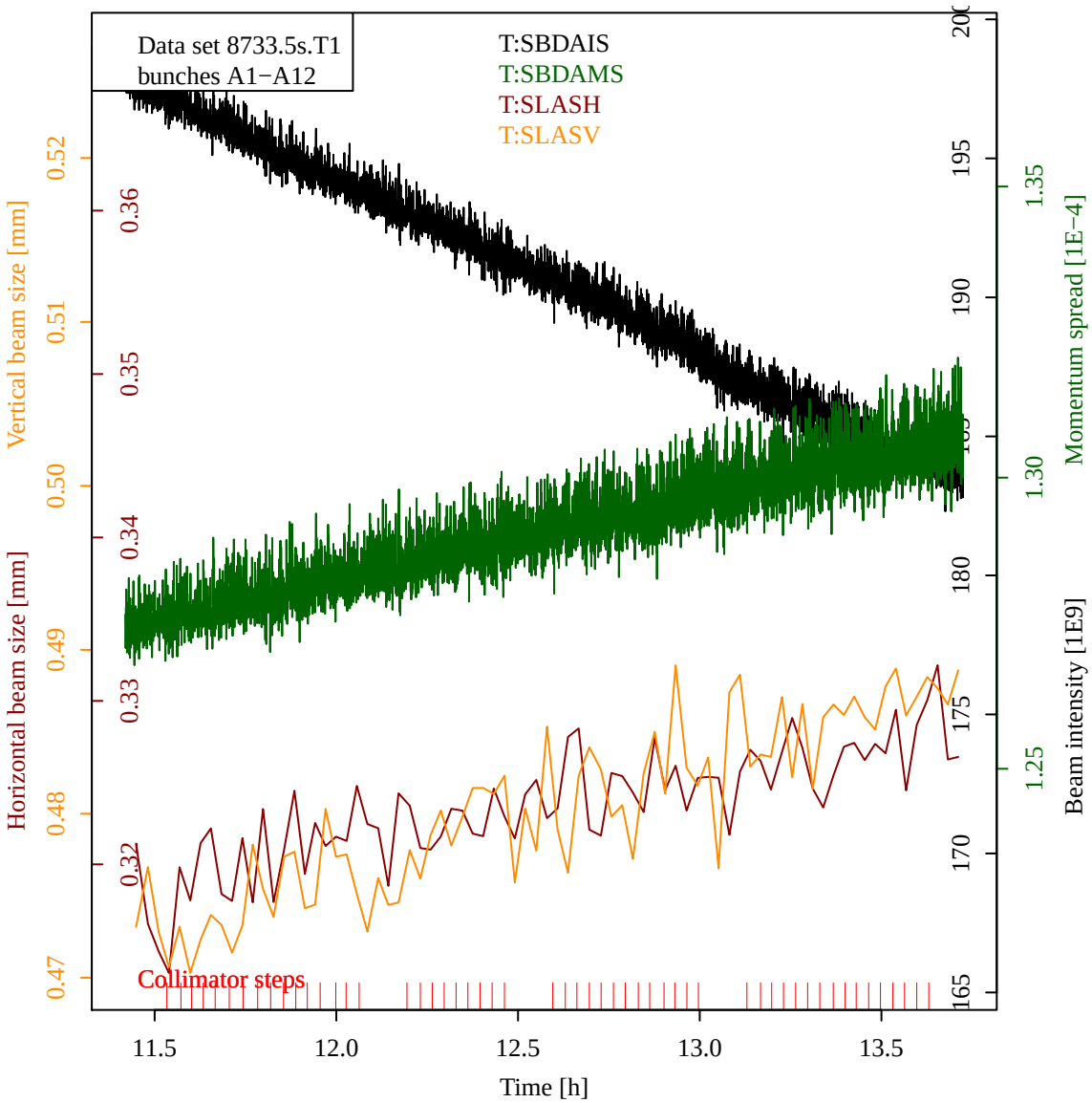
Does the outward step model work?

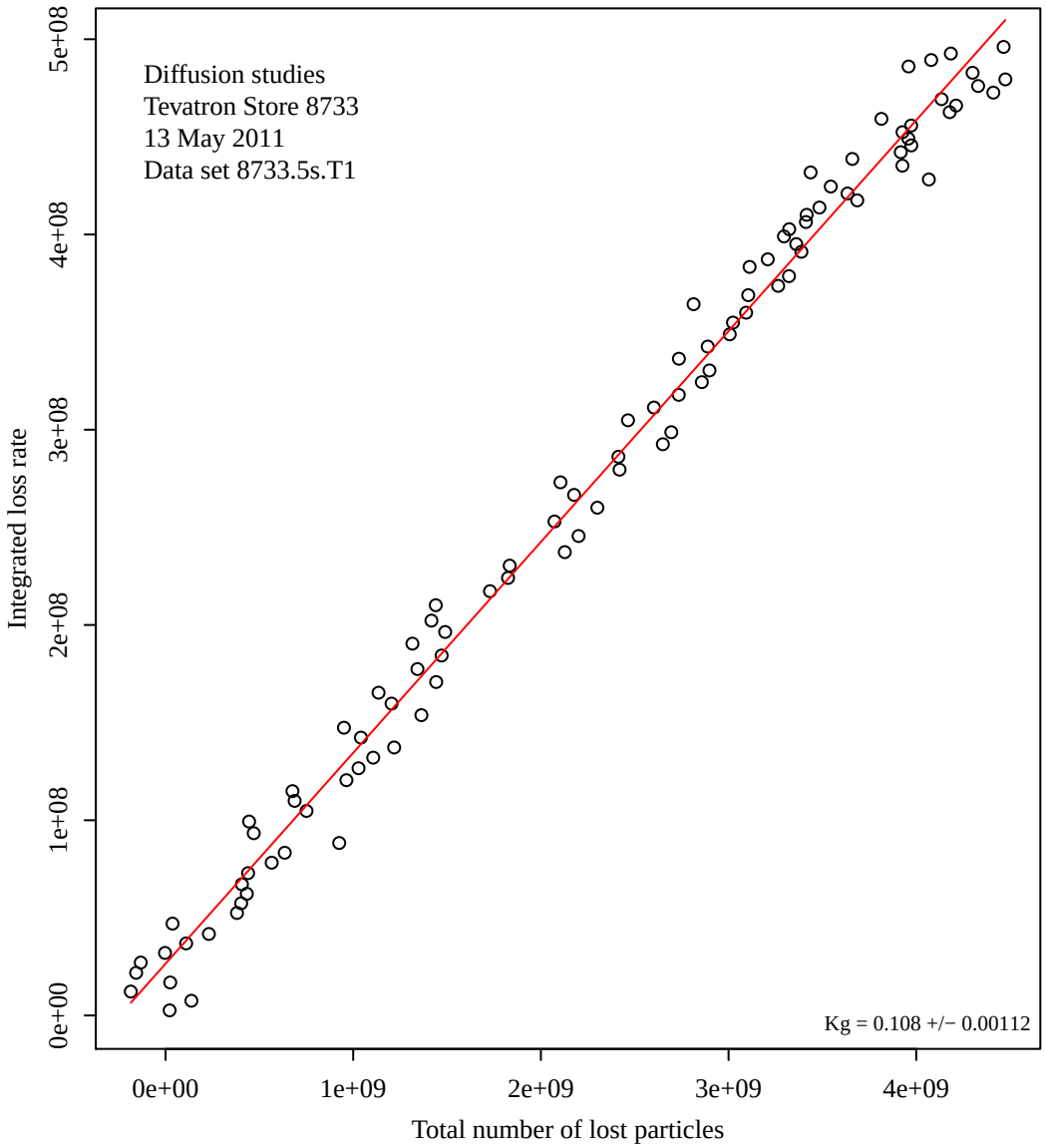
Comments on beam population measurements.

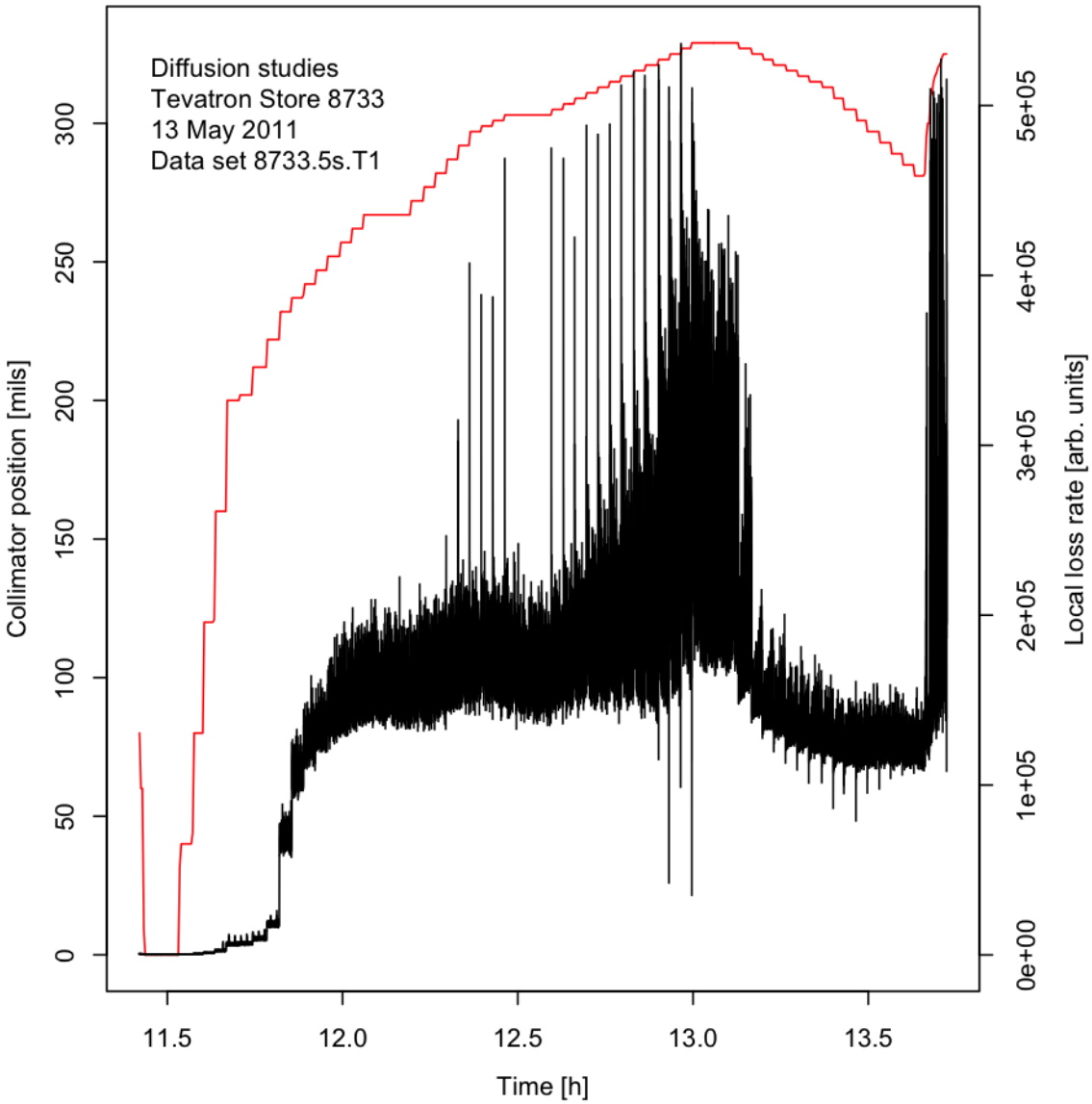
Appendix A: Diagnostic plots

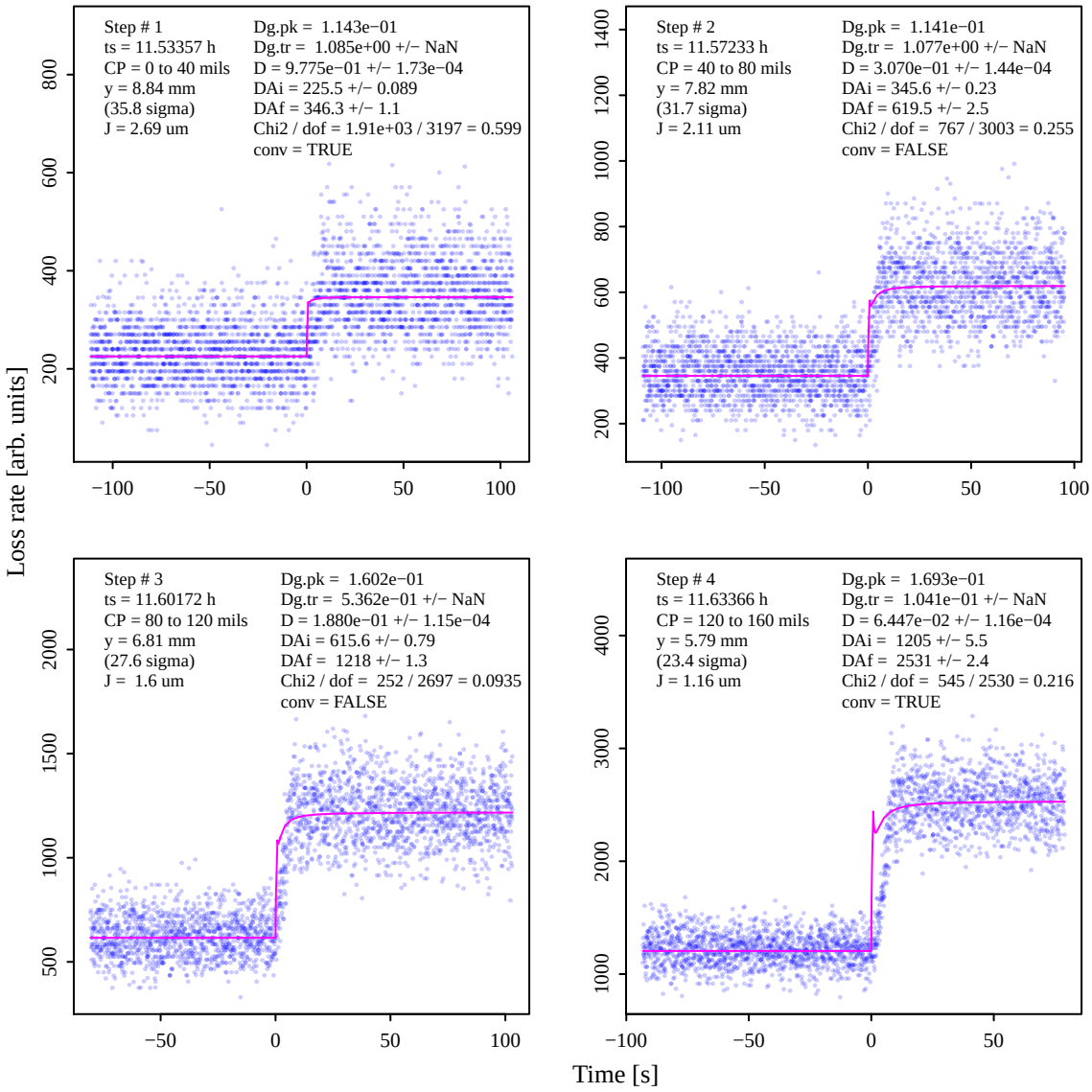
1. Data set 8733.5s.T1

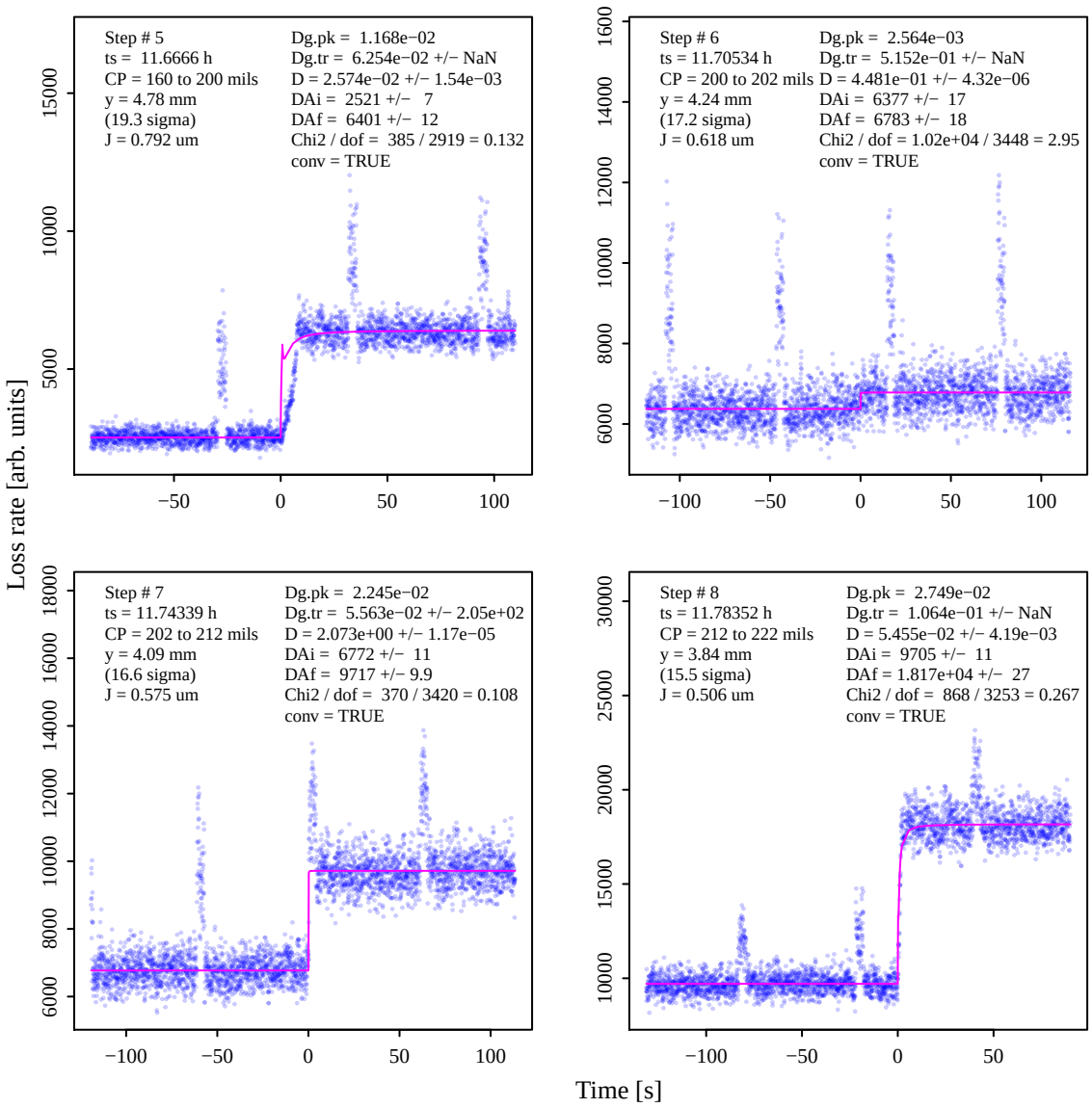


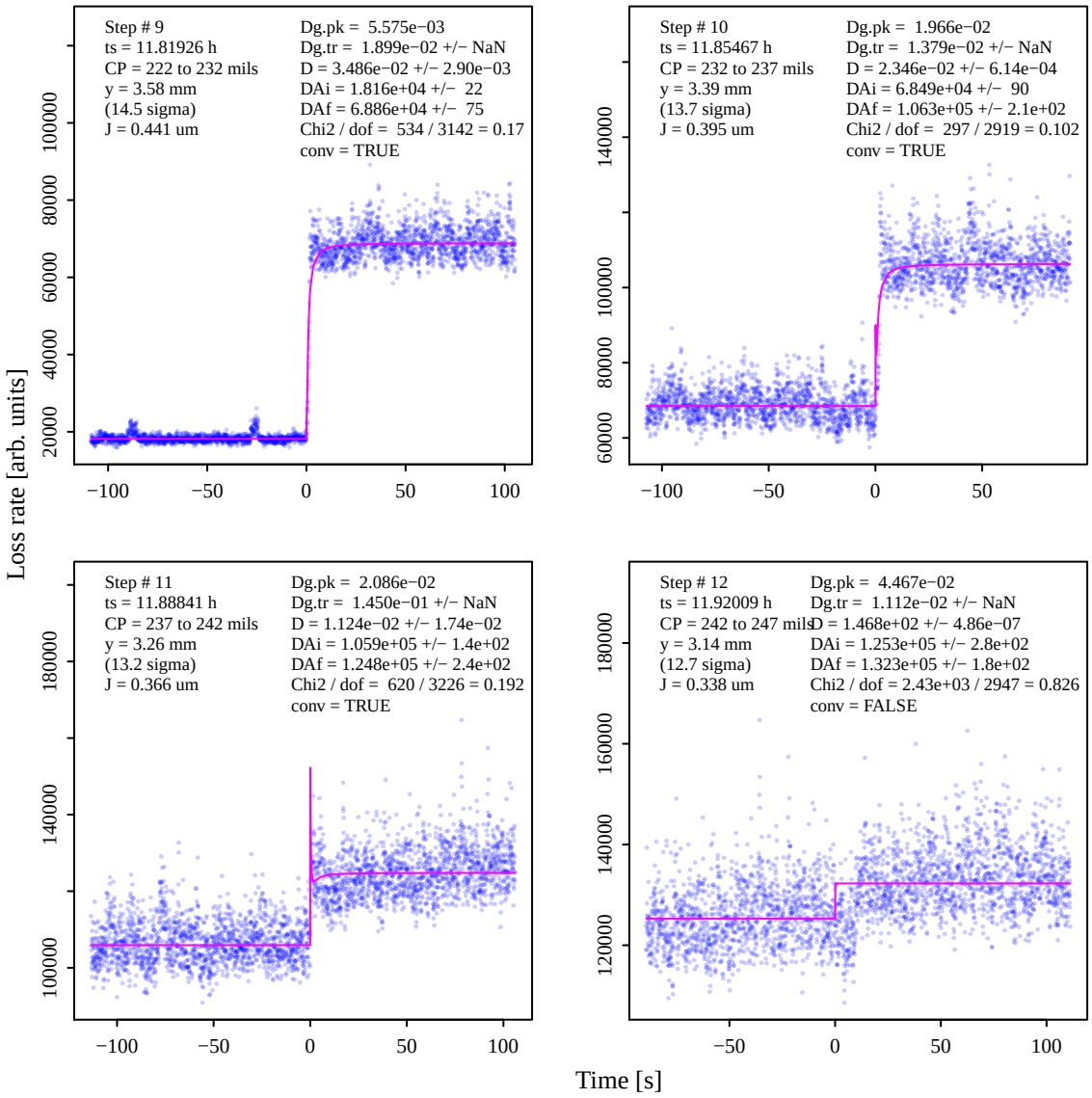

Data set 8733.5s.T1
bunches A1−A12
T:SBDAIS
T:SBDAMS
T:SLASH
T:SLASV
Vertical beam size [mm]
Horizontal beam size [mm]
Momentum spread [1E−4]
Beam intensity [1E9]
Collimator steps
Time [h]
11.5
12.0
12.5
13.0
13.5
0.47
0.48
0.49
0.50
0.51
0.52
0.32
0.33
0.34
0.35
0.36
165
170
175
180
190
195
1.25
1.30
1.35

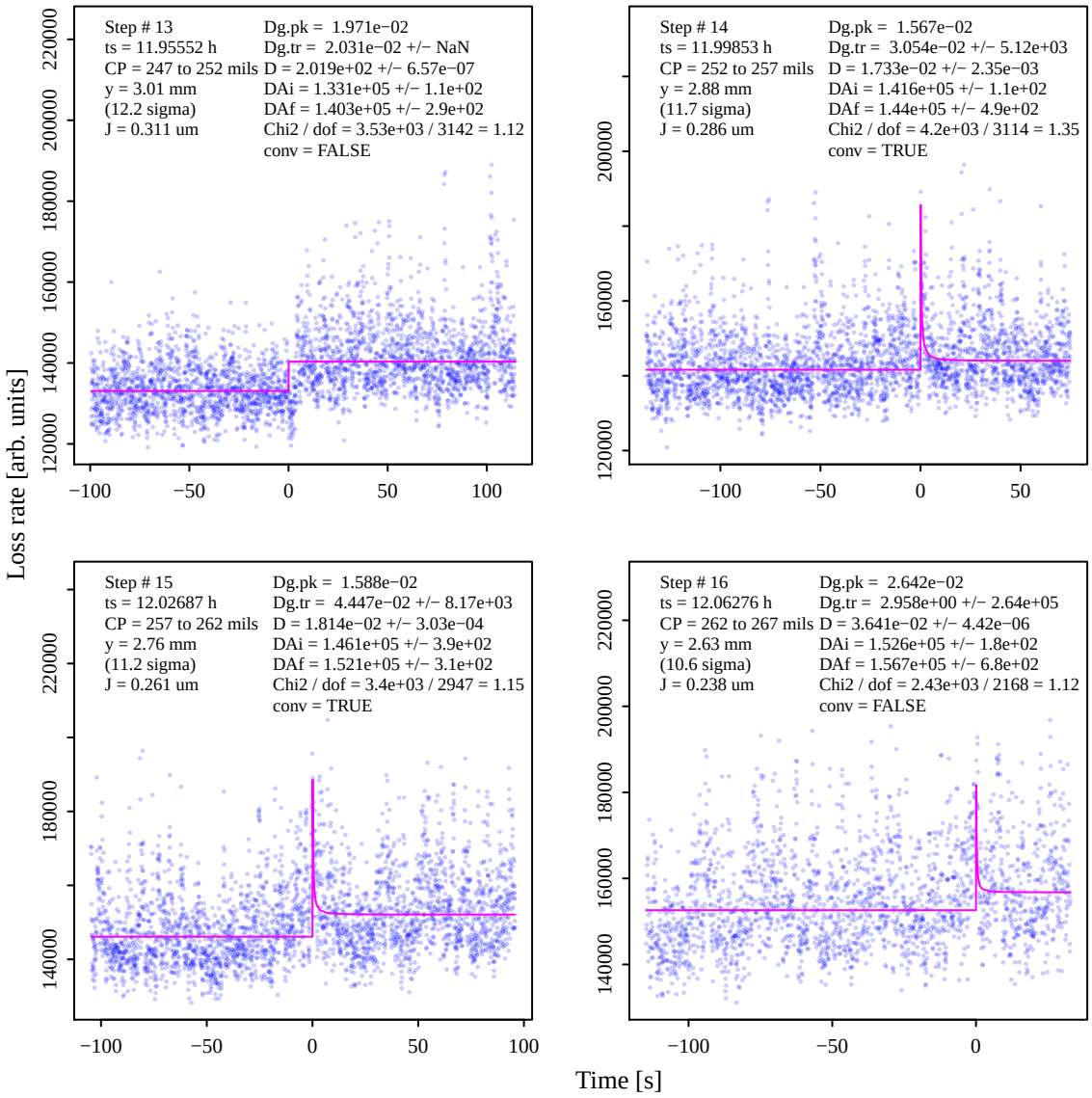

Diffusion studies
Tevatron Store 8733
13 May 2011
Data set 8733.5s.T1
Kg = 0.108 +/− 0.00112
Integrated loss rate
0e+00
1e+08
2e+08
3e+08
4e+08
5e+08
0e+00
1e+09
2e+09
3e+09
4e+09
Total number of lost particles

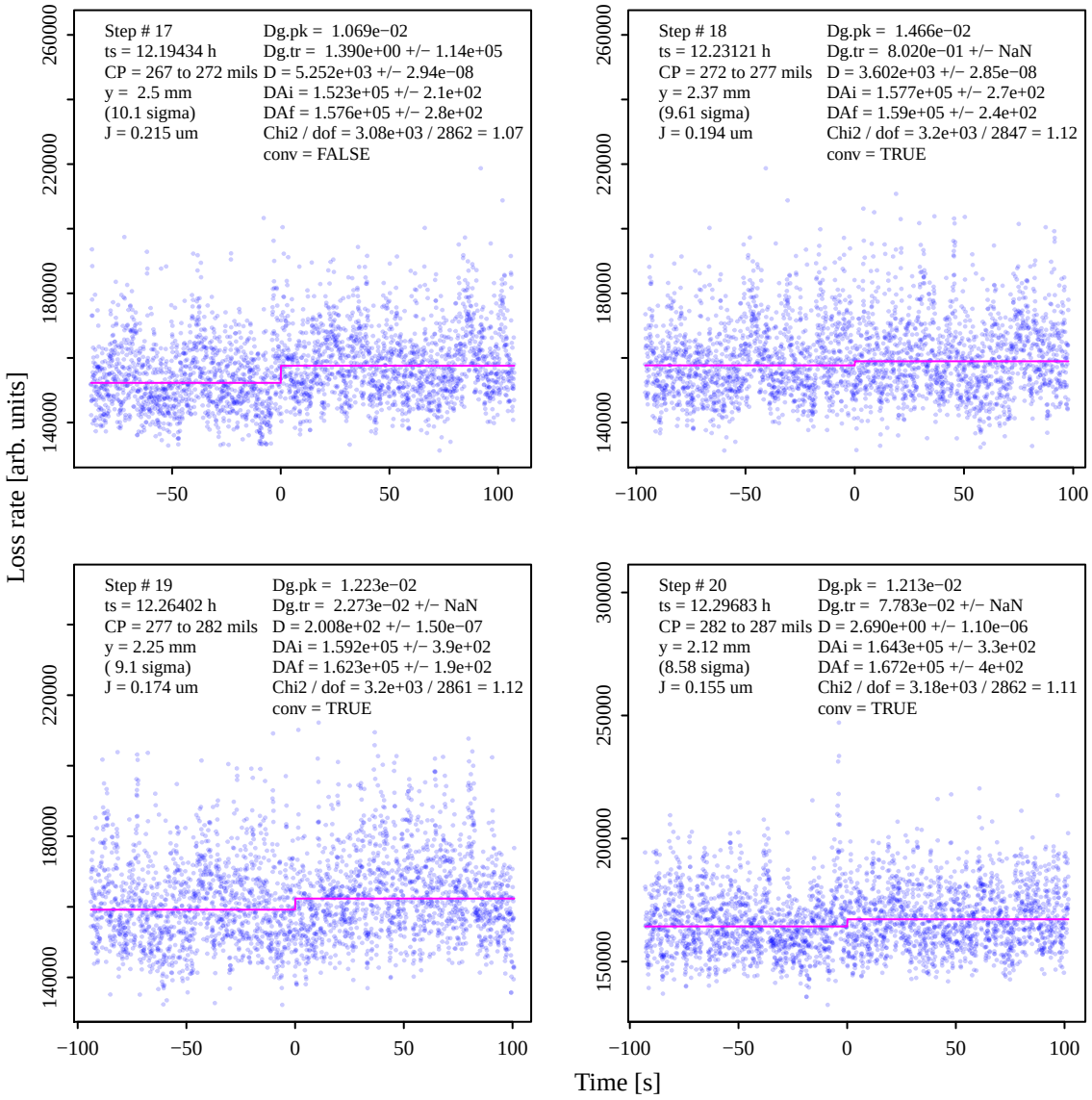

Diffusion studies
Tevatron Store 8733
13 May 2011
Data set 8733.5s.T1
Collimator position [mils]
0
50
100
150
200
250
300
Local loss rate [arb. units]
0e+00
1e+05
2e+05
3e+05
4e+05
5e+05
11.5
12.0
12.5
13.0
13.5
Time [h]

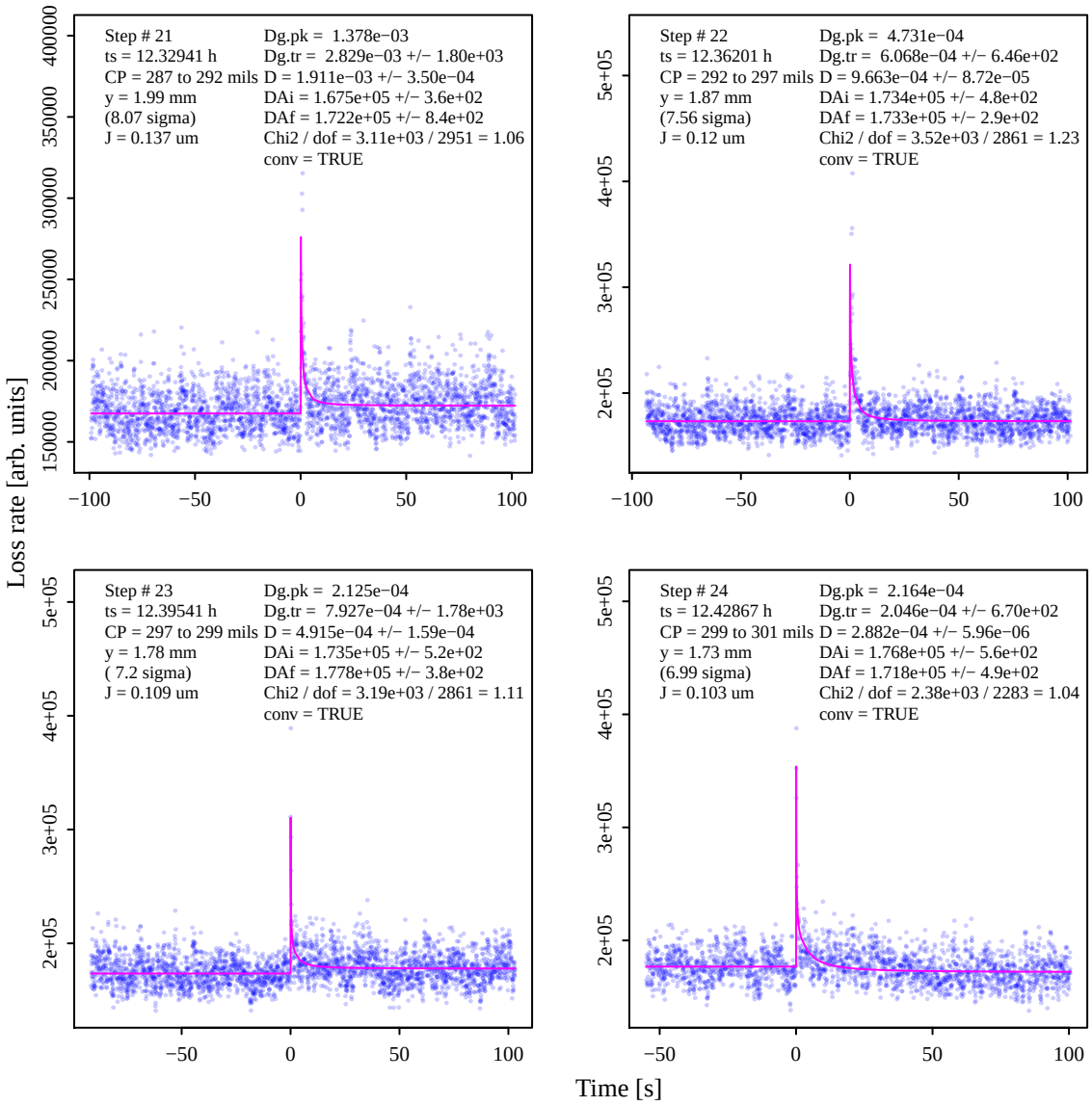

Step # 1
ts = 11.53357 h
CP = 0 to 40 mils
y = 8.84 mm
(35.8 sigma)
J = 2.69 um
Dg.pk = 1.143e−01
Dg.tr = 1.085e+00 +/− NaN
D = 9.775e−01 +/− 1.73e−04
DAi = 225.5 +/− 0.089
DAf = 346.3 +/− 1.1
Chi2 / dof = 1.91e+03 / 3197 = 0.599
conv = TRUE
Step # 2
ts = 11.57233 h
CP = 40 to 80 mils
y = 7.82 mm
(31.7 sigma)
J = 2.11 um
Dg.pk = 1.141e−01
Dg.tr = 1.077e+00 +/− NaN
D = 3.070e−01 +/− 1.44e−04
DAi = 345.6 +/− 0.23
DAf = 619.5 +/− 2.5
Chi2 / dof = 767 / 3003 = 0.255
conv = FALSE
Step # 3
ts = 11.60172 h
CP = 80 to 120 mils
y = 6.81 mm
(27.6 sigma)
J = 1.6 um
Dg.pk = 1.602e−01
Dg.tr = 5.362e−01 +/− NaN
D = 1.880e−01 +/− 1.15e−04
DAi = 615.6 +/− 0.79
DAf = 1218 +/− 1.3
Chi2 / dof = 252 / 2697 = 0.0935
conv = FALSE
Step # 4
ts = 11.63366 h
CP = 120 to 160 mils
y = 5.79 mm
(23.4 sigma)
J = 1.16 um
Dg.pk = 1.693e−01
Dg.tr = 1.041e−01 +/− NaN
D = 6.447e−02 +/− 1.16e−04
DAi = 1205 +/− 5.5
DAf = 2531 +/− 2.4
Chi2 / dof = 545 / 2530 = 0.216
conv = TRUE
Loss rate [arb. units]
Time [s]

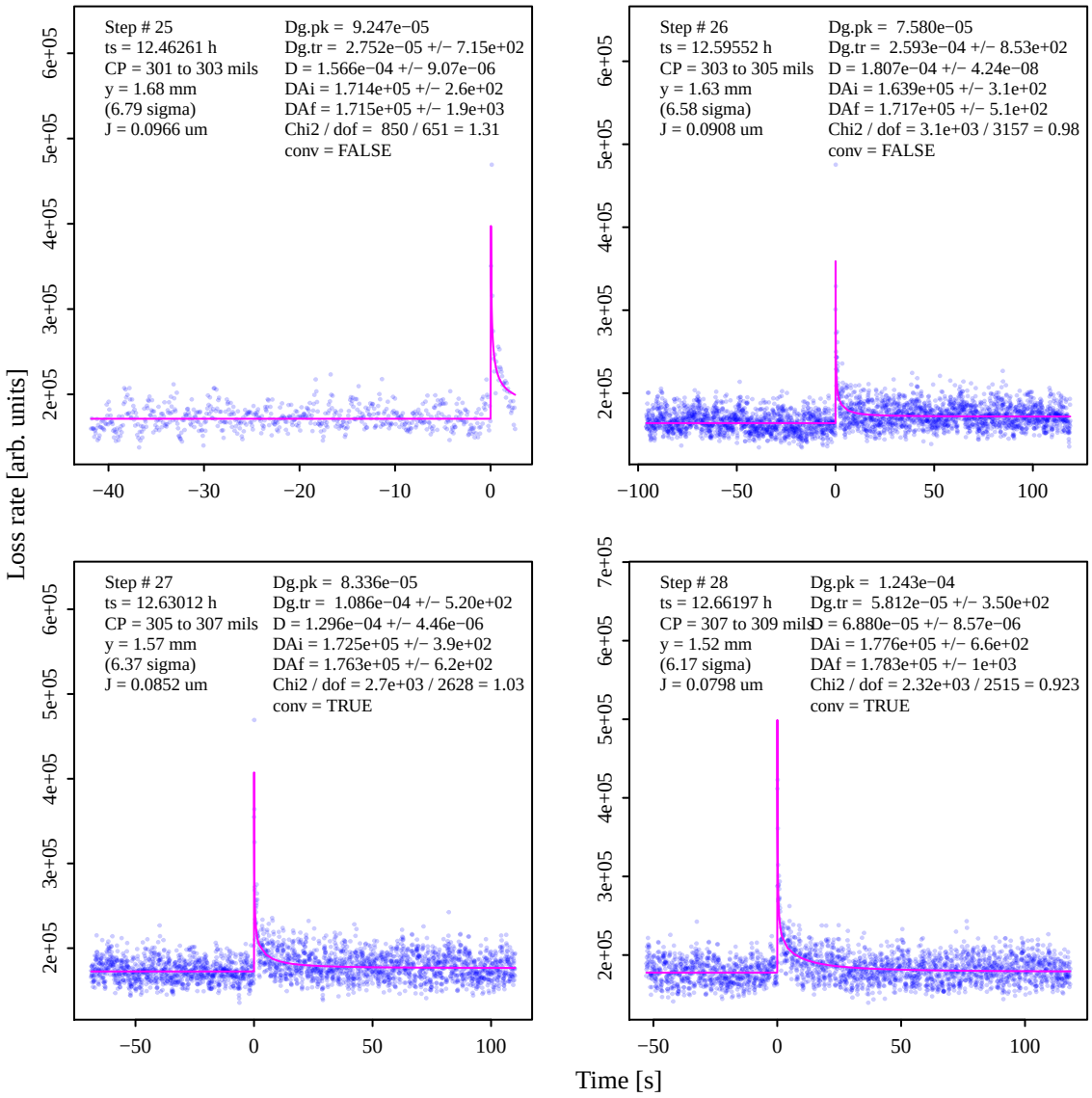

Step # 5
ts = 11.6666 h
CP = 160 to 200 mils
y = 4.78 mm
(19.3 sigma)
J = 0.792 um
Dg.pk = 1.168e−02
Dg.tr = 6.254e−02 +/− NaN
D = 2.574e−02 +/− 1.54e−03
DAi = 2521 +/− 7
DAf = 6401 +/− 12
Chi2 / dof = 385 / 2919 = 0.132
conv = TRUE
Step # 6
ts = 11.70534 h
CP = 200 to 202 mils
y = 4.24 mm
(17.2 sigma)
J = 0.618 um
Dg.pk = 2.564e−03
Dg.tr = 5.152e−01 +/− NaN
D = 4.481e−01 +/− 4.32e−06
DAi = 6377 +/− 17
DAf = 6783 +/− 18
Chi2 / dof = 1.02e+04 / 3448 = 2.95
conv = TRUE
Step # 7
ts = 11.74339 h
CP = 202 to 212 mils
y = 4.09 mm
(16.6 sigma)
J = 0.575 um
Dg.pk = 2.245e−02
Dg.tr = 5.563e−02 +/− 2.05e+02
D = 2.073e+00 +/− 1.17e−05
DAi = 6772 +/− 11
DAf = 9717 +/− 9.9
Chi2 / dof = 370 / 3420 = 0.108
conv = TRUE
Step # 8
ts = 11.78352 h
CP = 212 to 222 mils
y = 3.84 mm
(15.5 sigma)
J = 0.506 um
Dg.pk = 2.749e−02
Dg.tr = 1.064e−01 +/− NaN
D = 5.455e−02 +/− 4.19e−03
DAi = 9705 +/− 11
DAf = 1.817e+04 +/− 27
Chi2 / dof = 868 / 3253 = 0.267
conv = TRUE
Loss rate [arb. units]
Time [s]

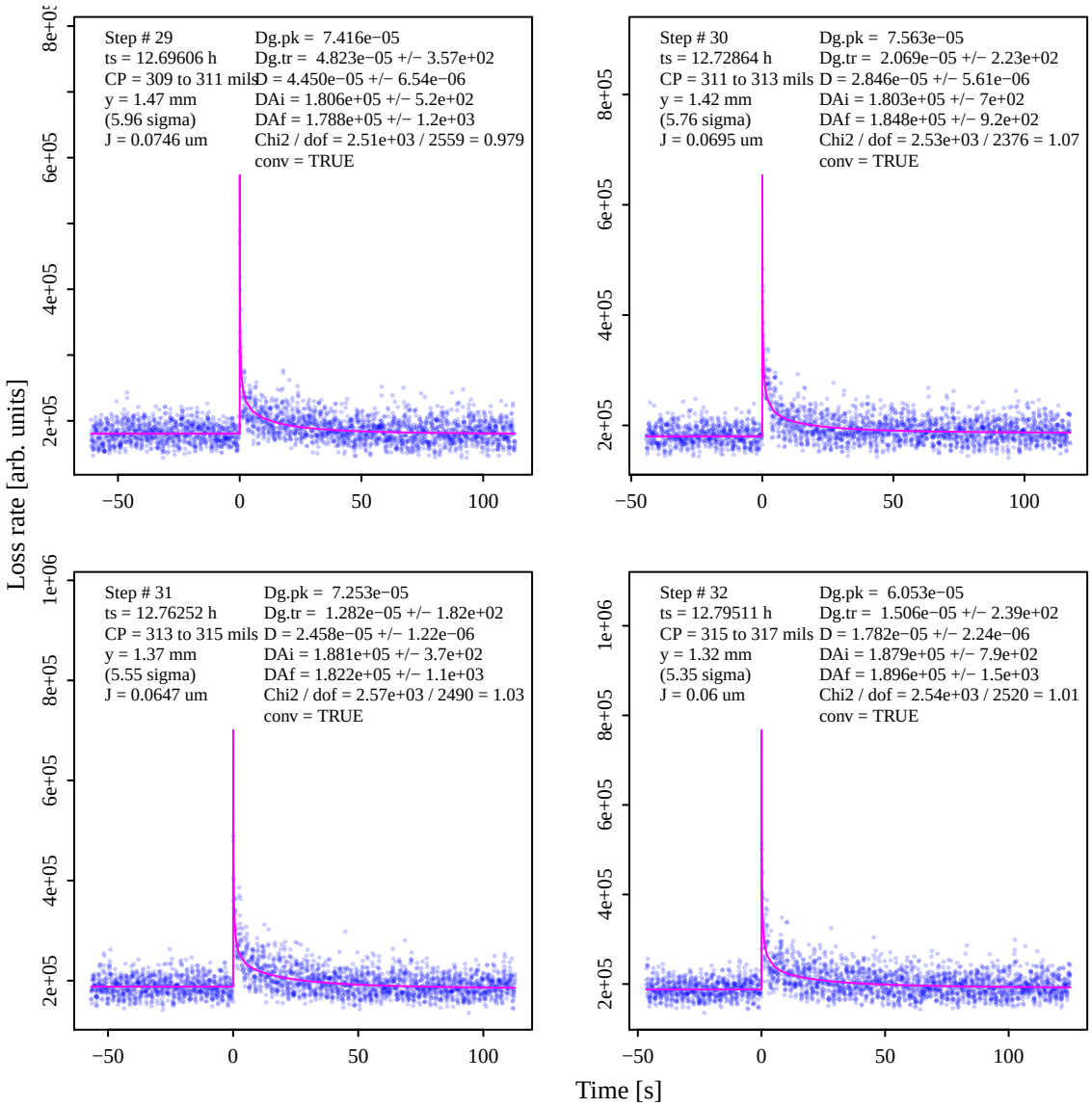

Step # 9
ts = 11.81926 h
CP = 222 to 232 mils
y = 3.58 mm
(14.5 sigma)
J = 0.441 um
Dg.pk = 5.575e−03
Dg.tr = 1.899e−02 +/− NaN
D = 3.486e−02 +/− 2.90e−03
DAi = 1.816e+04 +/− 22
DAf = 6.886e+04 +/− 75
Chi2 / dof = 534 / 3142 = 0.17
conv = TRUE
Step # 10
ts = 11.85467 h
CP = 232 to 237 mils
y = 3.39 mm
(13.7 sigma)
J = 0.395 um
Dg.pk = 1.966e−02
Dg.tr = 1.379e−02 +/− NaN
D = 2.346e−02 +/− 6.14e−04
DAi = 6.849e+04 +/− 90
DAf = 1.063e+05 +/− 2.1e+02
Chi2 / dof = 297 / 2919 = 0.102
conv = TRUE
Step # 11
ts = 11.88841 h
CP = 237 to 242 mils
y = 3.26 mm
(13.2 sigma)
J = 0.366 um
Dg.pk = 2.086e−02
Dg.tr = 1.450e−01 +/− NaN
D = 1.124e−02 +/− 1.74e−02
DAi = 1.059e+05 +/− 1.4e+02
DAf = 1.248e+05 +/− 2.4e+02
Chi2 / dof = 620 / 3226 = 0.192
conv = TRUE
Step # 12
ts = 11.92009 h
CP = 242 to 247 mils
y = 3.14 mm
(12.7 sigma)
J = 0.338 um
Dg.pk = 4.467e−02
Dg.tr = 1.112e−02 +/− NaN
D = 1.468e+02 +/− 4.86e−07
DAi = 1.253e+05 +/− 2.8e+02
DAf = 1.323e+05 +/− 1.8e+02
Chi2 / dof = 2.43e+03 / 2947 = 0.826
conv = FALSE
Loss rate [arb. units]
Time [s]

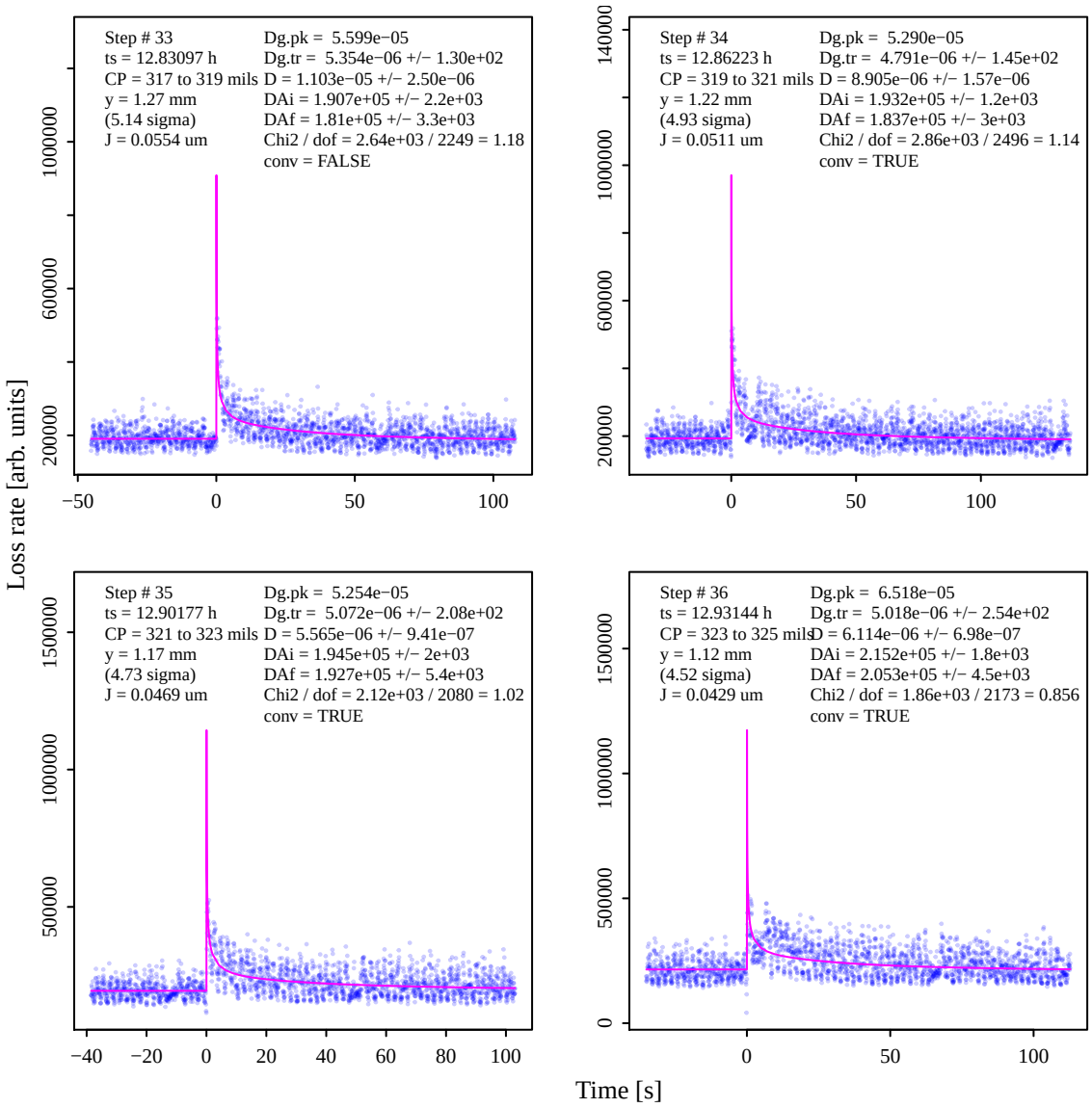

Step # 13
ts = 11.95552 h
CP = 247 to 252 mils
y = 3.01 mm
(12.2 sigma)
J = 0.311 um
Dg.pk = 1.971e−02
Dg.tr = 2.031e−02 +/− NaN
D = 2.019e+02 +/− 6.57e−07
DAi = 1.331e+05 +/− 1.1e+02
DAf = 1.403e+05 +/− 2.9e+02
Chi2 / dof = 3.53e+03 / 3142 = 1.12
conv = FALSE
Step # 14
ts = 11.99853 h
CP = 252 to 257 mils
y = 2.88 mm
(11.7 sigma)
J = 0.286 um
Dg.pk = 1.567e−02
Dg.tr = 3.054e−02 +/− 5.12e+03
D = 1.733e−02 +/− 2.35e−03
DAi = 1.416e+05 +/− 1.1e+02
DAf = 1.44e+05 +/− 4.9e+02
Chi2 / dof = 4.2e+03 / 3114 = 1.35
conv = TRUE
Step # 15
ts = 12.02687 h
CP = 257 to 262 mils
y = 2.76 mm
(11.2 sigma)
J = 0.261 um
Dg.pk = 1.588e−02
Dg.tr = 4.447e−02 +/− 8.17e+03
D = 1.814e−02 +/− 3.03e−04
DAi = 1.461e+05 +/− 3.9e+02
DAf = 1.521e+05 +/− 3.1e+02
Chi2 / dof = 3.4e+03 / 2947 = 1.15
conv = TRUE
Step # 16
ts = 12.06276 h
CP = 262 to 267 mils
y = 2.63 mm
(10.6 sigma)
J = 0.238 um
Dg.pk = 2.642e−02
Dg.tr = 2.958e+00 +/− 2.64e+05
D = 3.641e−02 +/− 4.42e−06
DAi = 1.526e+05 +/− 1.8e+02
DAf = 1.567e+05 +/− 6.8e+02
Chi2 / dof = 2.43e+03 / 2168 = 1.12
conv = FALSE
Loss rate [arb. units]
Time [s]

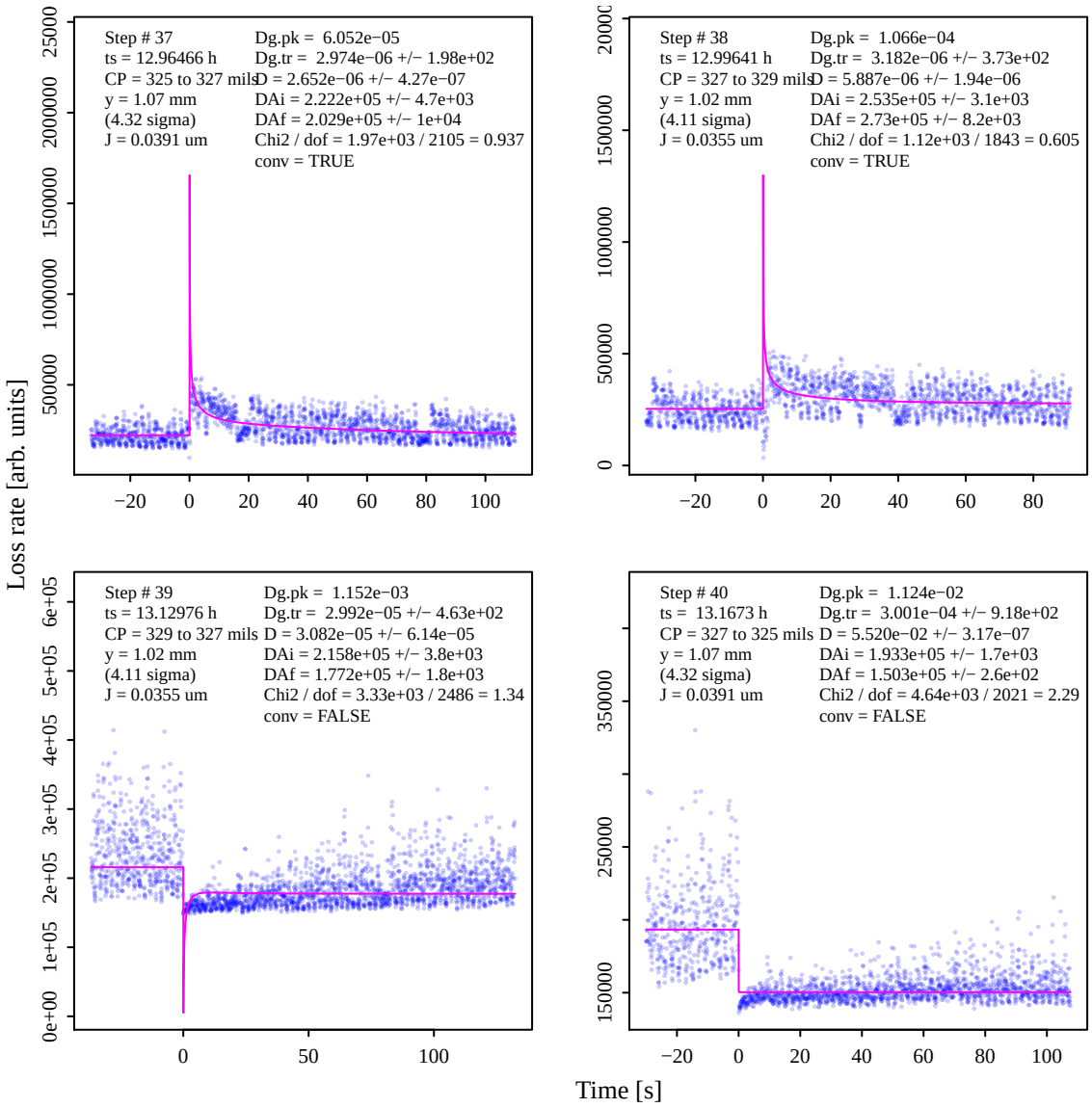

Step # 17
ts = 12.19434 h
CP = 267 to 272 mils
y = 2.5 mm
(10.1 sigma)
J = 0.215 um
Dg.pk = 1.069e−02
Dg.tr = 1.390e+00 +/− 1.14e+05
D = 5.252e+03 +/− 2.94e−08
DAi = 1.523e+05 +/− 2.1e+02
DAf = 1.576e+05 +/− 2.8e+02
Chi2 / dof = 3.08e+03 / 2862 = 1.07
conv = FALSE
Step # 18
ts = 12.23121 h
CP = 272 to 277 mils
y = 2.37 mm
(9.61 sigma)
J = 0.194 um
Dg.pk = 1.466e−02
Dg.tr = 8.020e−01 +/− NaN
D = 3.602e+03 +/− 2.85e−08
DAi = 1.577e+05 +/− 2.7e+02
DAf = 1.59e+05 +/− 2.4e+02
Chi2 / dof = 3.2e+03 / 2847 = 1.12
conv = TRUE
Step # 19
ts = 12.26402 h
CP = 277 to 282 mils
y = 2.25 mm
(9.1 sigma)
J = 0.174 um
Dg.pk = 1.223e−02
Dg.tr = 2.273e−02 +/− NaN
D = 2.008e+02 +/− 1.50e−07
DAi = 1.592e+05 +/− 3.9e+02
DAf = 1.623e+05 +/− 1.9e+02
Chi2 / dof = 3.2e+03 / 2861 = 1.12
conv = TRUE
Step # 20
ts = 12.29683 h
CP = 282 to 287 mils
y = 2.12 mm
(8.58 sigma)
J = 0.155 um
Dg.pk = 1.213e−02
Dg.tr = 7.783e−02 +/− NaN
D = 2.690e+00 +/− 1.10e−06
DAi = 1.643e+05 +/− 3.3e+02
DAf = 1.672e+05 +/− 4e+02
Chi2 / dof = 3.18e+03 / 2862 = 1.11
conv = TRUE
Loss rate [arb. units]
Time [s]

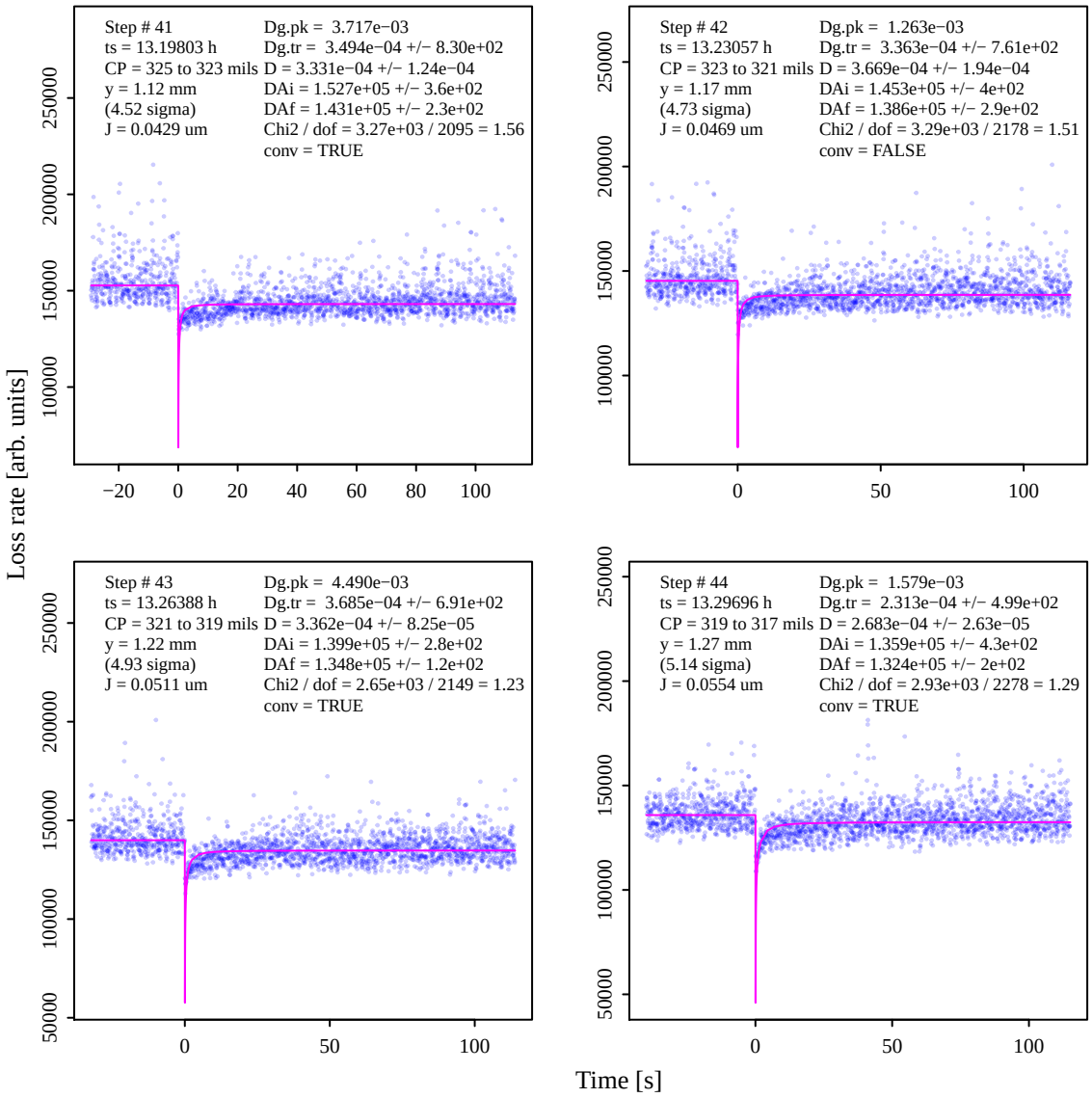

Step # 21
ts = 12.32941 h
CP = 287 to 292 mils
y = 1.99 mm
(8.07 sigma)
J = 0.137 um
Dg.pk = 1.378e−03
Dg.tr = 2.829e−03 +/− 1.80e+03
D = 1.911e−03 +/− 3.50e−04
DAi = 1.675e+05 +/− 3.6e+02
DAf = 1.722e+05 +/− 8.4e+02
Chi2 / dof = 3.11e+03 / 2951 = 1.06
conv = TRUE
Step # 22
ts = 12.36201 h
CP = 292 to 297 mils
y = 1.87 mm
(7.56 sigma)
J = 0.12 um
Dg.pk = 4.731e−04
Dg.tr = 6.068e−04 +/− 6.46e+02
D = 9.663e−04 +/− 8.72e−05
DAi = 1.734e+05 +/− 4.8e+02
DAf = 1.733e+05 +/− 2.9e+02
Chi2 / dof = 3.52e+03 / 2861 = 1.23
conv = TRUE
Step # 23
ts = 12.39541 h
CP = 297 to 299 mils
y = 1.78 mm
(7.2 sigma)
J = 0.109 um
Dg.pk = 2.125e−04
Dg.tr = 7.927e−04 +/− 1.78e+03
D = 4.915e−04 +/− 1.59e−04
DAi = 1.735e+05 +/− 5.2e+02
DAf = 1.778e+05 +/− 3.8e+02
Chi2 / dof = 3.19e+03 / 2861 = 1.11
conv = TRUE
Step # 24
ts = 12.42867 h
CP = 299 to 301 mils
y = 1.73 mm
(6.99 sigma)
J = 0.103 um
Dg.pk = 2.164e−04
Dg.tr = 2.046e−04 +/− 6.70e+02
D = 2.882e−04 +/− 5.96e−06
DAi = 1.768e+05 +/− 5.6e+02
DAf = 1.718e+05 +/− 4.9e+02
Chi2 / dof = 2.38e+03 / 2283 = 1.04
conv = TRUE
Loss rate [arb. units]
Time [s]

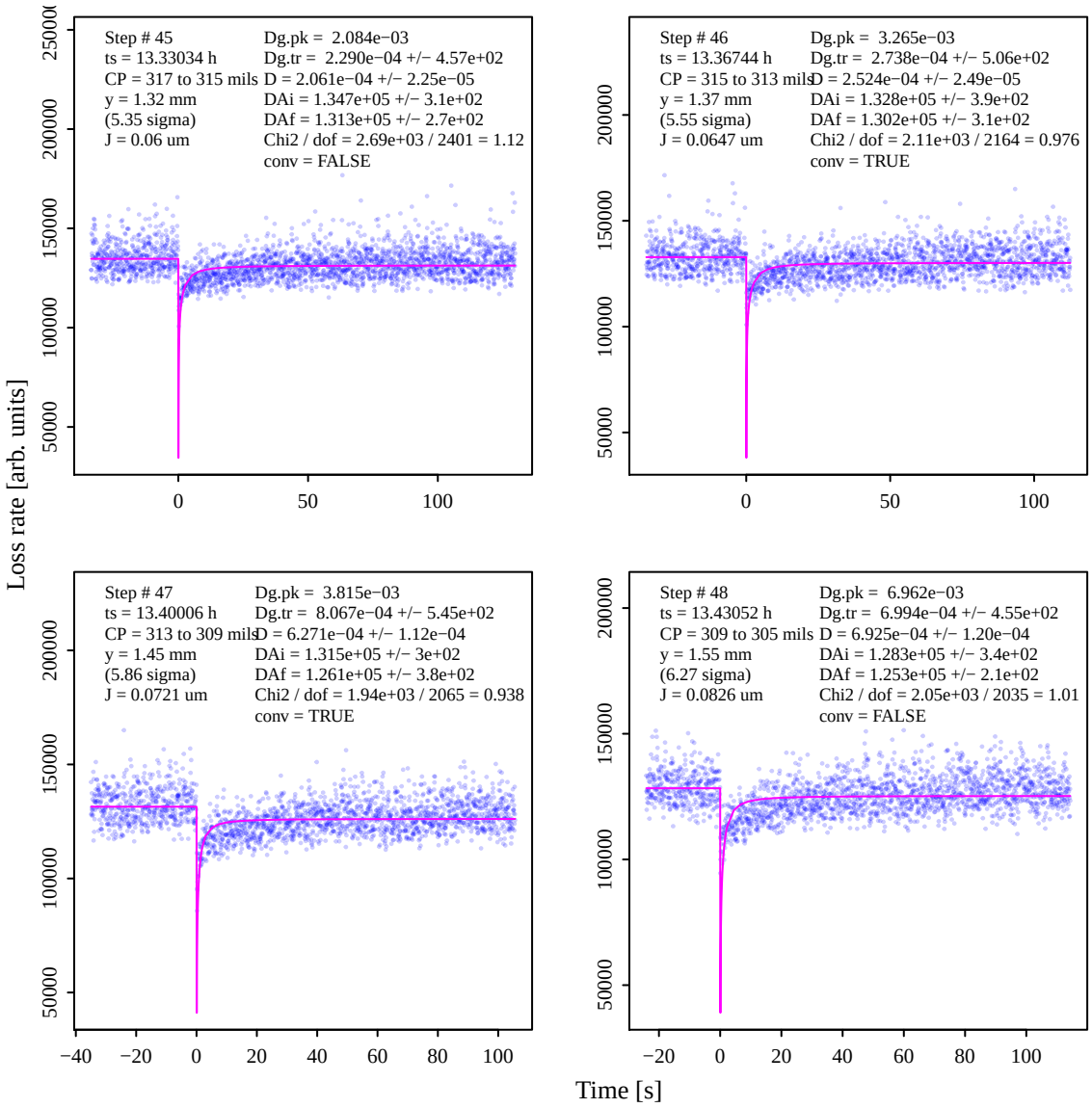

Step # 25
ts = 12.46261 h
CP = 301 to 303 mils
y = 1.68 mm
(6.79 sigma)
J = 0.0966 um
Dg.pk = 9.247e−05
Dg.tr = 2.752e−05 +/− 7.15e+02
D = 1.566e−04 +/− 9.07e−06
DAi = 1.714e+05 +/− 2.6e+02
DAf = 1.715e+05 +/− 1.9e+03
Chi2 / dof = 850 / 651 = 1.31
conv = FALSE
Step # 26
ts = 12.59552 h
CP = 303 to 305 mils
y = 1.63 mm
(6.58 sigma)
J = 0.0908 um
Dg.pk = 7.580e−05
Dg.tr = 2.593e−04 +/− 8.53e+02
D = 1.807e−04 +/− 4.24e−08
DAi = 1.639e+05 +/− 3.1e+02
DAf = 1.717e+05 +/− 5.1e+02
Chi2 / dof = 3.1e+03 / 3157 = 0.98
conv = FALSE
Step # 27
ts = 12.63012 h
CP = 305 to 307 mils
y = 1.57 mm
(6.37 sigma)
J = 0.0852 um
Dg.pk = 8.336e−05
Dg.tr = 1.086e−04 +/− 5.20e+02
D = 1.296e−04 +/− 4.46e−06
DAi = 1.725e+05 +/− 3.9e+02
DAf = 1.763e+05 +/− 6.2e+02
Chi2 / dof = 2.7e+03 / 2628 = 1.03
conv = TRUE
Step # 28
ts = 12.66197 h
CP = 307 to 309 mils
y = 1.52 mm
(6.17 sigma)
J = 0.0798 um
Dg.pk = 1.243e−04
Dg.tr = 5.812e−05 +/− 3.50e+02
D = 6.880e−05 +/− 8.57e−06
DAi = 1.776e+05 +/− 6.6e+02
DAf = 1.783e+05 +/− 1e+03
Chi2 / dof = 2.32e+03 / 2515 = 0.923
conv = TRUE
Loss rate [arb. units]
Time [s]

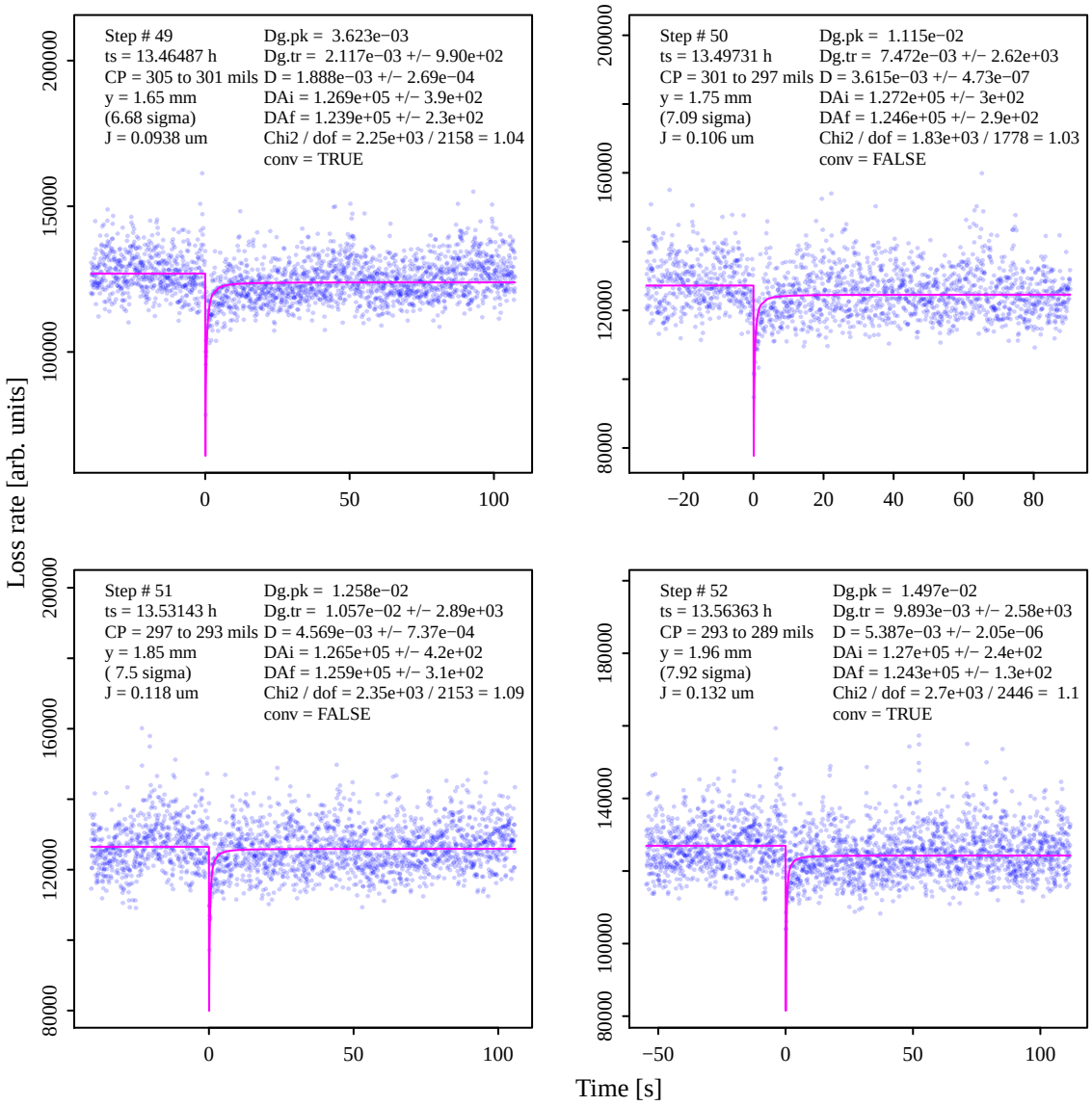

Step # 29
ts = 12.69606 h
CP = 309 to 311 mils
y = 1.47 mm
(5.96 sigma)
J = 0.0746 um
Dg.pk = 7.416e−05
Dg.tr = 4.823e−05 +/− 3.57e+02
D = 4.450e−05 +/− 6.54e−06
DAi = 1.806e+05 +/− 5.2e+02
DAf = 1.788e+05 +/− 1.2e+03
Chi2 / dof = 2.51e+03 / 2559 = 0.979
conv = TRUE
Step # 30
ts = 12.72864 h
CP = 311 to 313 mils
y = 1.42 mm
(5.76 sigma)
J = 0.0695 um
Dg.pk = 7.563e−05
Dg.tr = 2.069e−05 +/− 2.23e+02
D = 2.846e−05 +/− 5.61e−06
DAi = 1.803e+05 +/− 7e+02
DAf = 1.848e+05 +/− 9.2e+02
Chi2 / dof = 2.53e+03 / 2376 = 1.07
conv = TRUE
Step # 31
ts = 12.76252 h
CP = 313 to 315 mils
y = 1.37 mm
(5.55 sigma)
J = 0.0647 um
Dg.pk = 7.253e−05
Dg.tr = 1.282e−05 +/− 1.82e+02
D = 2.458e−05 +/− 1.22e−06
DAi = 1.881e+05 +/− 3.7e+02
DAf = 1.822e+05 +/− 1.1e+03
Chi2 / dof = 2.57e+03 / 2490 = 1.03
conv = TRUE
Step # 32
ts = 12.79511 h
CP = 315 to 317 mils
y = 1.32 mm
(5.35 sigma)
J = 0.06 um
Dg.pk = 6.053e−05
Dg.tr = 1.506e−05 +/− 2.39e+02
D = 1.782e−05 +/− 2.24e−06
DAi = 1.879e+05 +/− 7.9e+02
DAf = 1.896e+05 +/− 1.5e+03
Chi2 / dof = 2.54e+03 / 2520 = 1.01
conv = TRUE
Loss rate [arb. units]
Time [s]

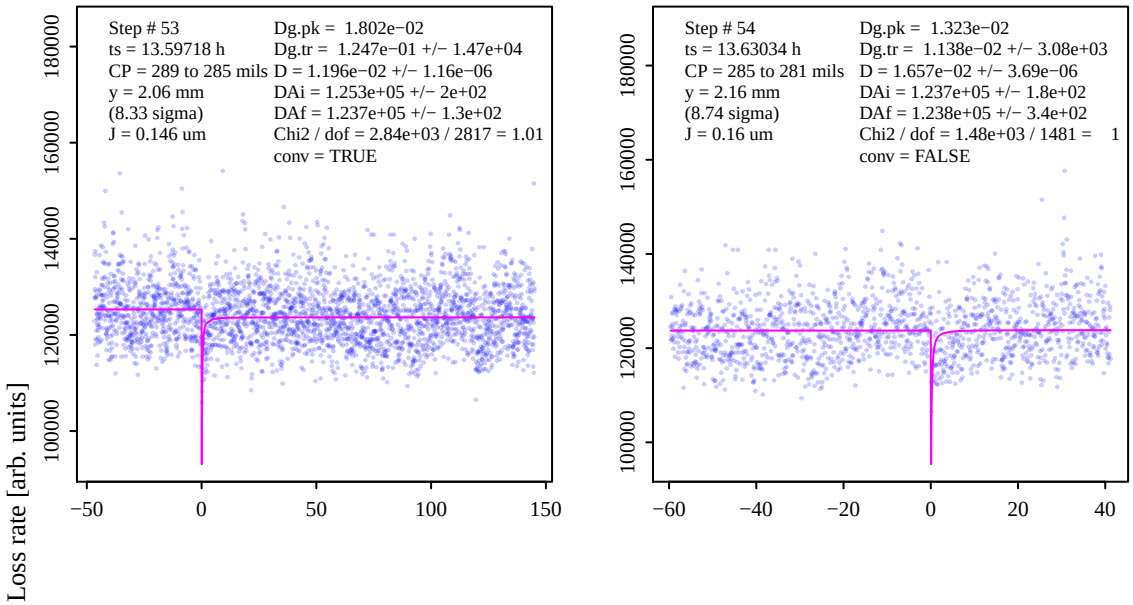

Step # 33
ts = 12.83097 h
CP = 317 to 319 mils
y = 1.27 mm
(5.14 sigma)
J = 0.0554 um
Dg.pk = 5.599e−05
Dg.tr = 5.354e−06 +/− 1.30e+02
D = 1.103e−05 +/− 2.50e−06
DAi = 1.907e+05 +/− 2.2e+03
DAf = 1.81e+05 +/− 3.3e+03
Chi2 / dof = 2.64e+03 / 2249 = 1.18
conv = FALSE
Step # 34
ts = 12.86223 h
CP = 319 to 321 mils
y = 1.22 mm
(4.93 sigma)
J = 0.0511 um
Dg.pk = 5.290e−05
Dg.tr = 4.791e−06 +/− 1.45e+02
D = 8.905e−06 +/− 1.57e−06
DAi = 1.932e+05 +/− 1.2e+03
DAf = 1.837e+05 +/− 3e+03
Chi2 / dof = 2.86e+03 / 2496 = 1.14
conv = TRUE
Step # 35
ts = 12.90177 h
CP = 321 to 323 mils
y = 1.17 mm
(4.73 sigma)
J = 0.0469 um
Dg.pk = 5.254e−05
Dg.tr = 5.072e−06 +/− 2.08e+02
D = 5.565e−06 +/− 9.41e−07
DAi = 1.945e+05 +/− 2e+03
DAf = 1.927e+05 +/− 5.4e+03
Chi2 / dof = 2.12e+03 / 2080 = 1.02
conv = TRUE
Step # 36
ts = 12.93144 h
CP = 323 to 325 mils
y = 1.12 mm
(4.52 sigma)
J = 0.0429 um
Dg.pk = 6.518e−05
Dg.tr = 5.018e−06 +/− 2.54e+02
D = 6.114e−06 +/− 6.98e−07
DAi = 2.152e+05 +/− 1.8e+03
DAf = 2.053e+05 +/− 4.5e+03
Chi2 / dof = 1.86e+03 / 2173 = 0.856
conv = TRUE
Loss rate [arb. units]
Time [s]

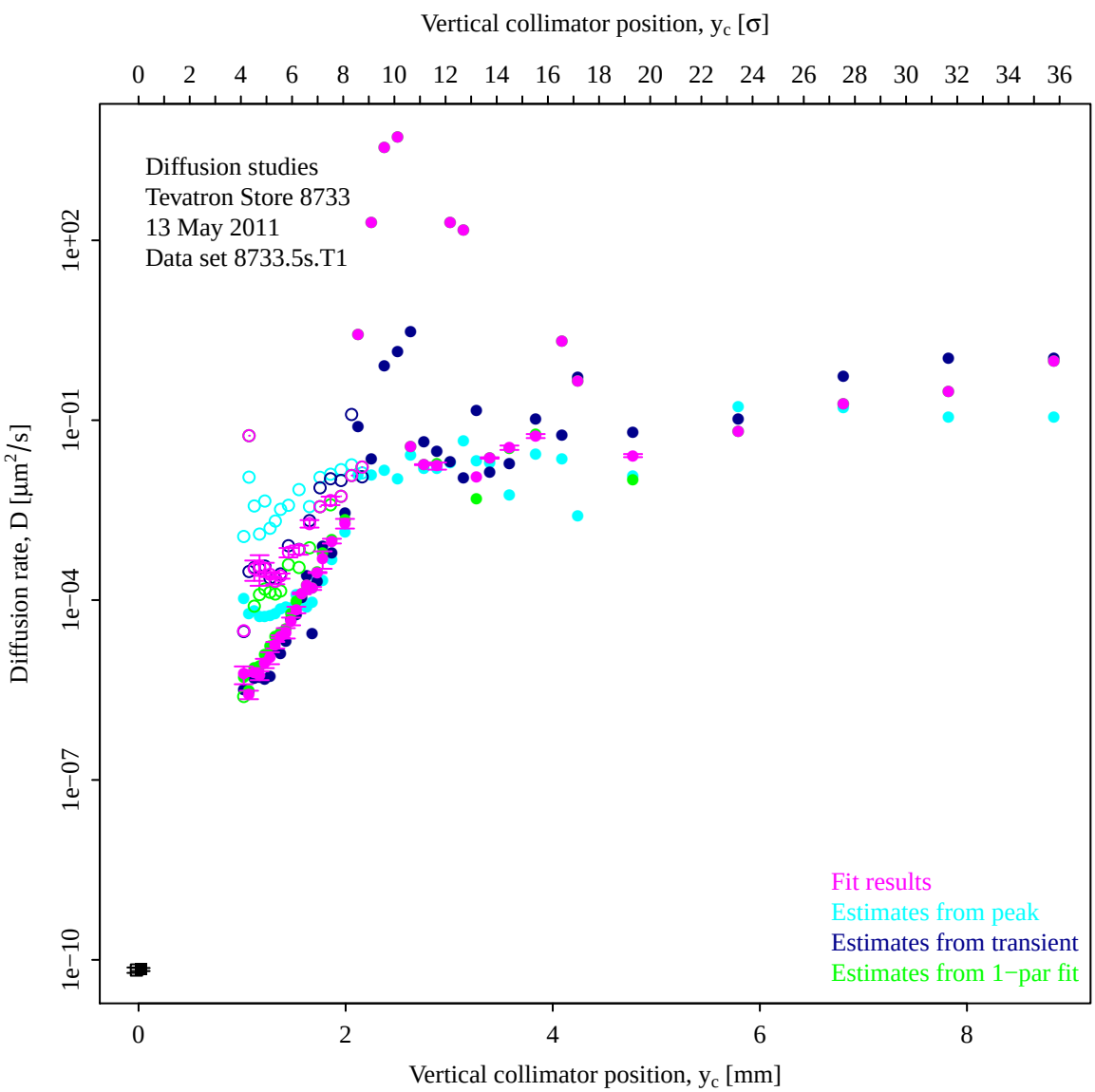

Step # 37
ts = 12.96466 h
CP = 325 to 327 mils
y = 1.07 mm
(4.32 sigma)
J = 0.0391 um
Dg.pk = 6.052e−05
Dg.tr = 2.974e−06 +/− 1.98e+02
D = 2.652e−06 +/− 4.27e−07
DAi = 2.222e+05 +/− 4.7e+03
DAf = 2.029e+05 +/− 1e+04
Chi2 / dof = 1.97e+03 / 2105 = 0.937
conv = TRUE
Step # 38
ts = 12.99641 h
CP = 327 to 329 mils
y = 1.02 mm
(4.11 sigma)
J = 0.0355 um
Dg.pk = 1.066e−04
Dg.tr = 3.182e−06 +/− 3.73e+02
D = 5.887e−06 +/− 1.94e−06
DAi = 2.535e+05 +/− 3.1e+03
DAf = 2.73e+05 +/− 8.2e+03
Chi2 / dof = 1.12e+03 / 1843 = 0.605
conv = TRUE
Step # 39
ts = 13.12976 h
CP = 329 to 327 mils
y = 1.02 mm
(4.11 sigma)
J = 0.0355 um
Dg.pk = 1.152e−03
Dg.tr = 2.992e−05 +/− 4.63e+02
D = 3.082e−05 +/− 6.14e−05
DAi = 2.158e+05 +/− 3.8e+03
DAf = 1.772e+05 +/− 1.8e+03
Chi2 / dof = 3.33e+03 / 2486 = 1.34
conv = FALSE
Step # 40
ts = 13.1673 h
CP = 327 to 325 mils
y = 1.07 mm
(4.32 sigma)
J = 0.0391 um
Dg.pk = 1.124e−02
Dg.tr = 3.001e−04 +/− 9.18e+02
D = 5.520e−02 +/− 3.17e−07
DAi = 1.933e+05 +/− 1.7e+03
DAf = 1.503e+05 +/− 2.6e+02
Chi2 / dof = 4.64e+03 / 2021 = 2.29
conv = FALSE
Loss rate [arb. units]
Time [s]

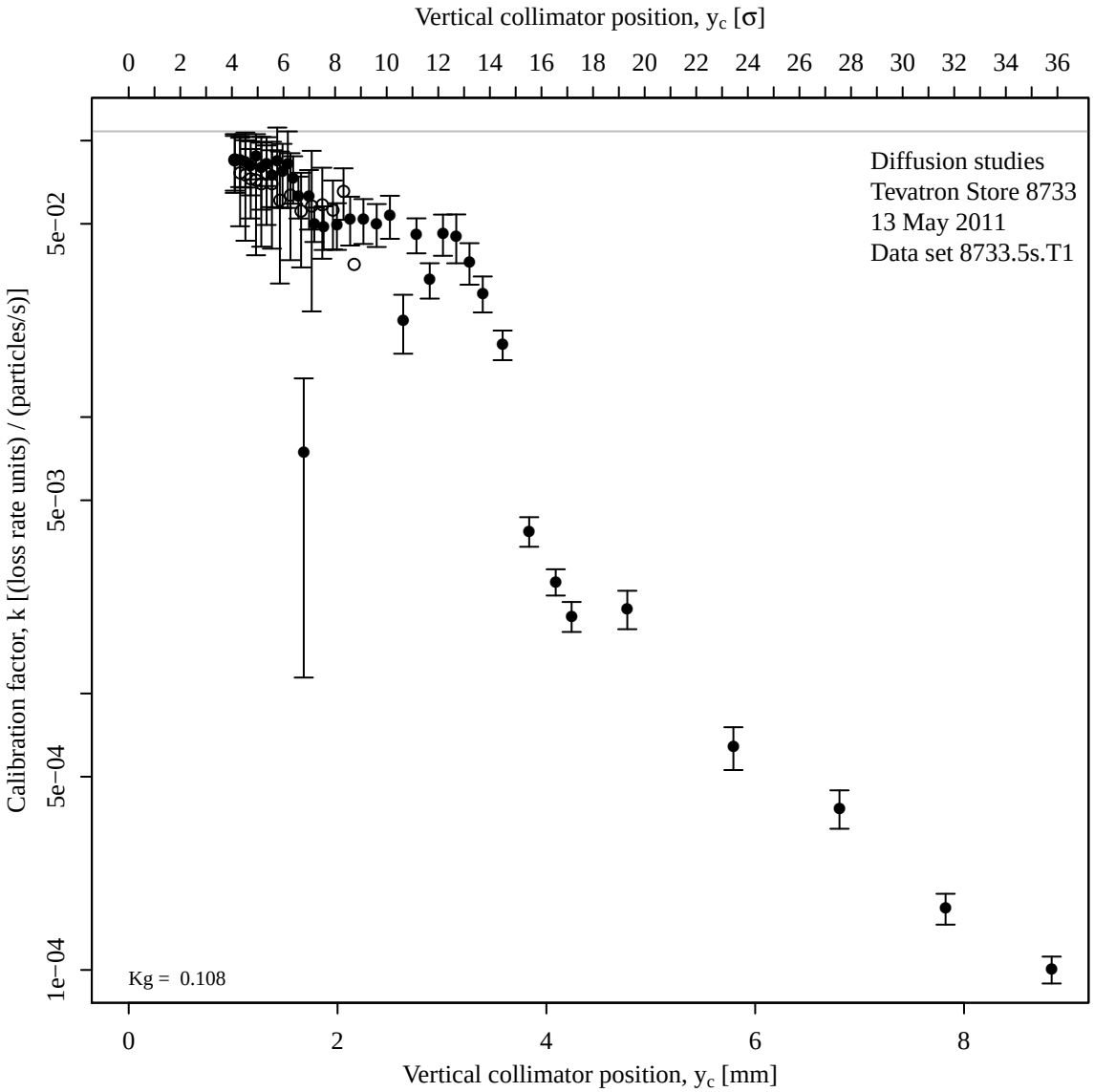

Step # 41
ts = 13.19803 h
CP = 325 to 323 mils
y = 1.12 mm
(4.52 sigma)
J = 0.0429 um
Dg.pk = 3.717e−03
Dg.tr = 3.494e−04 +/− 8.30e+02
D = 3.331e−04 +/− 1.24e−04
DAi = 1.527e+05 +/− 3.6e+02
DAf = 1.431e+05 +/− 2.3e+02
Chi2 / dof = 3.27e+03 / 2095 = 1.56
conv = TRUE
Step # 42
ts = 13.23057 h
CP = 323 to 321 mils
y = 1.17 mm
(4.73 sigma)
J = 0.0469 um
Dg.pk = 1.263e−03
Dg.tr = 3.363e−04 +/− 7.61e+02
D = 3.669e−04 +/− 1.94e−04
DAi = 1.453e+05 +/− 4e+02
DAf = 1.386e+05 +/− 2.9e+02
Chi2 / dof = 3.29e+03 / 2178 = 1.51
conv = FALSE
Step # 43
ts = 13.26388 h
CP = 321 to 319 mils
y = 1.22 mm
(4.93 sigma)
J = 0.0511 um
Dg.pk = 4.490e−03
Dg.tr = 3.685e−04 +/− 6.91e+02
D = 3.362e−04 +/− 8.25e−05
DAi = 1.399e+05 +/− 2.8e+02
DAf = 1.348e+05 +/− 1.2e+02
Chi2 / dof = 2.65e+03 / 2149 = 1.23
conv = TRUE
Step # 44
ts = 13.29696 h
CP = 319 to 317 mils
y = 1.27 mm
(5.14 sigma)
J = 0.0554 um
Dg.pk = 1.579e−03
Dg.tr = 2.313e−04 +/− 4.99e+02
D = 2.683e−04 +/− 2.63e−05
DAi = 1.359e+05 +/− 4.3e+02
DAf = 1.324e+05 +/− 2e+02
Chi2 / dof = 2.93e+03 / 2278 = 1.29
conv = TRUE
Loss rate [arb. units]
Time [s]

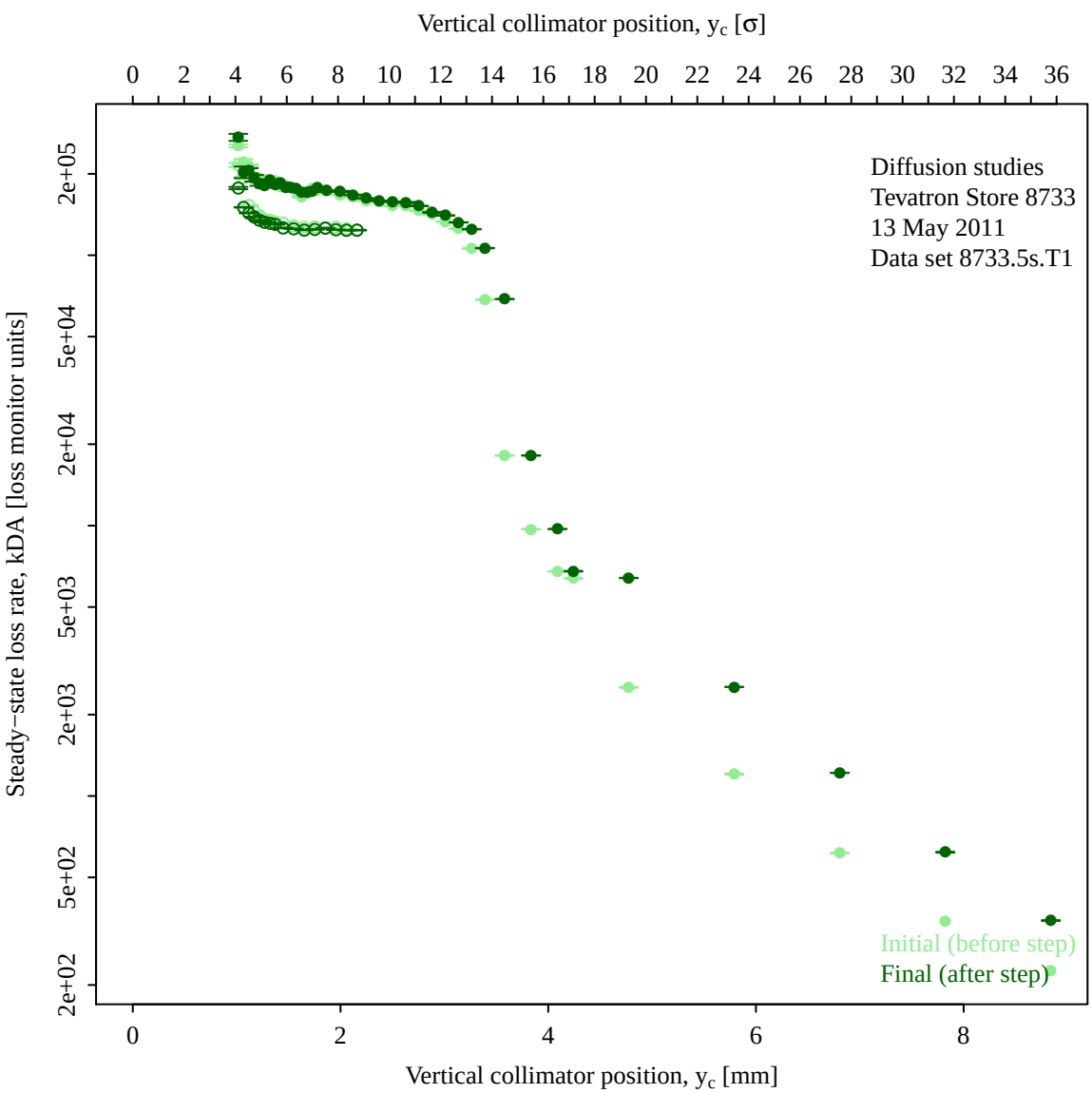

Step # 45
ts = 13.33034 h
CP = 317 to 315 mils
y = 1.32 mm
(5.35 sigma)
J = 0.06 um
Dg.pk = 2.084e−03
Dg.tr = 2.290e−04 +/− 4.57e+02
D = 2.061e−04 +/− 2.25e−05
DAi = 1.347e+05 +/− 3.1e+02
DAf = 1.313e+05 +/− 2.7e+02
Chi2 / dof = 2.69e+03 / 2401 = 1.12
conv = FALSE
Step # 46
ts = 13.36744 h
CP = 315 to 313 mils
y = 1.37 mm
(5.55 sigma)
J = 0.0647 um
Dg.pk = 3.265e−03
Dg.tr = 2.738e−04 +/− 5.06e+02
D = 2.524e−04 +/− 2.49e−05
DAi = 1.328e+05 +/− 3.9e+02
DAf = 1.302e+05 +/− 3.1e+02
Chi2 / dof = 2.11e+03 / 2164 = 0.976
conv = TRUE
Step # 47
ts = 13.40006 h
CP = 313 to 309 mils
y = 1.45 mm
(5.86 sigma)
J = 0.0721 um
Dg.pk = 3.815e−03
Dg.tr = 8.067e−04 +/− 5.45e+02
D = 6.271e−04 +/− 1.12e−04
DAi = 1.315e+05 +/− 3e+02
DAf = 1.261e+05 +/− 3.8e+02
Chi2 / dof = 1.94e+03 / 2065 = 0.938
conv = TRUE
Step # 48
ts = 13.43052 h
CP = 309 to 305 mils
y = 1.55 mm
(6.27 sigma)
J = 0.0826 um
Dg.pk = 6.962e−03
Dg.tr = 6.994e−04 +/− 4.55e+02
D = 6.925e−04 +/− 1.20e−04
DAi = 1.283e+05 +/− 3.4e+02
DAf = 1.253e+05 +/− 2.1e+02
Chi2 / dof = 2.05e+03 / 2035 = 1.01
conv = FALSE
Loss rate [arb. units]
Time [s]

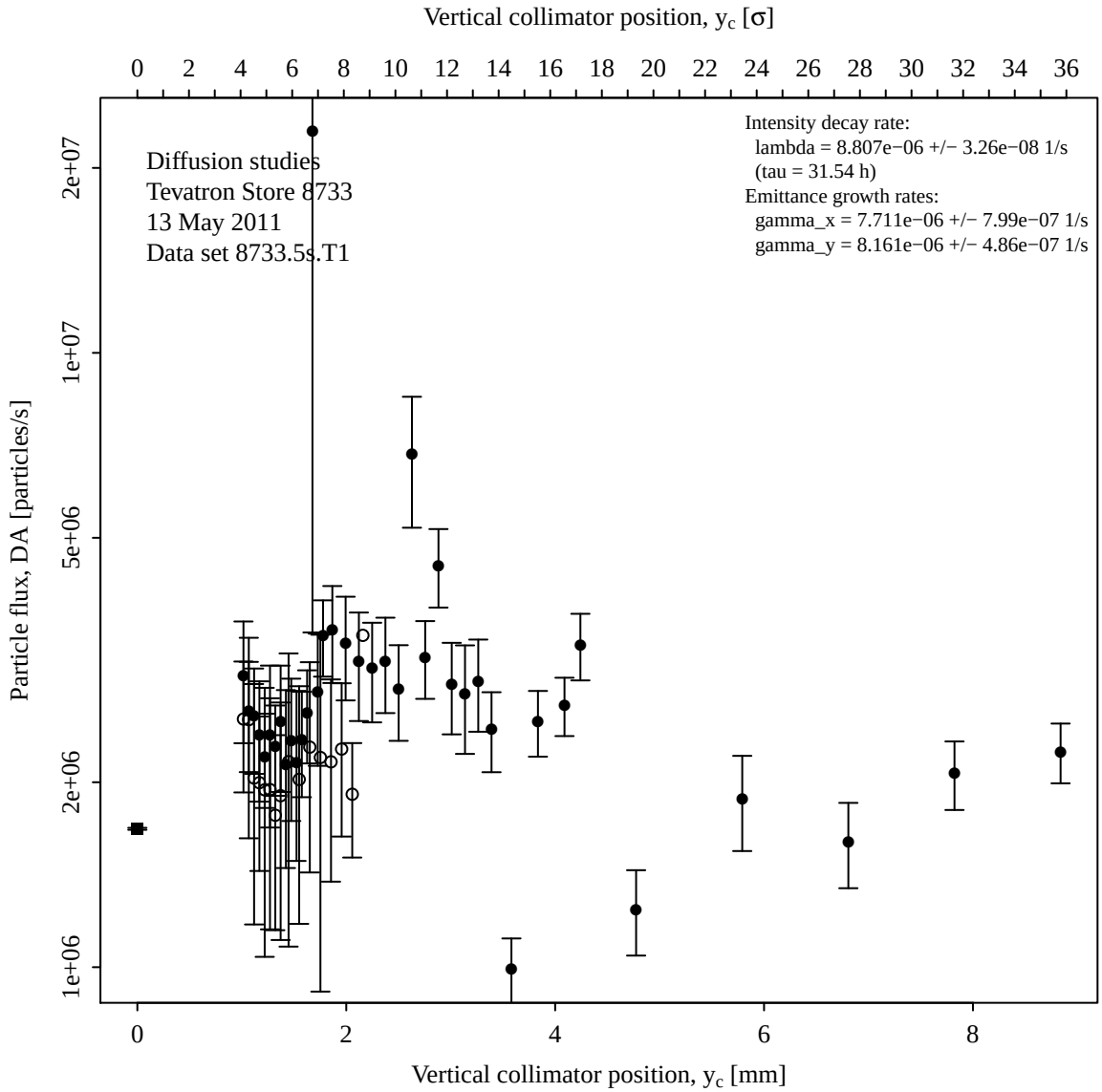

Step # 49
ts = 13.46487 h
CP = 305 to 301 mils
y = 1.65 mm
(6.68 sigma)
J = 0.0938 um
Dg.pk = 3.623e−03
Dg.tr = 2.117e−03 +/− 9.90e+02
D = 1.888e−03 +/− 2.69e−04
DAi = 1.269e+05 +/− 3.9e+02
DAf = 1.239e+05 +/− 2.3e+02
Chi2 / dof = 2.25e+03 / 2158 = 1.04
conv = TRUE
Step # 50
ts = 13.49731 h
CP = 301 to 297 mils
y = 1.75 mm
(7.09 sigma)
J = 0.106 um
Dg.pk = 1.115e−02
Dg.tr = 7.472e−03 +/− 2.62e+03
D = 3.615e−03 +/− 4.73e−07
DAi = 1.272e+05 +/− 3e+02
DAf = 1.246e+05 +/− 2.9e+02
Chi2 / dof = 1.83e+03 / 1778 = 1.03
conv = FALSE
Step # 51
ts = 13.53143 h
CP = 297 to 293 mils
y = 1.85 mm
(7.5 sigma)
J = 0.118 um
Dg.pk = 1.258e−02
Dg.tr = 1.057e−02 +/− 2.89e+03
D = 4.569e−03 +/− 7.37e−04
DAi = 1.265e+05 +/− 4.2e+02
DAf = 1.259e+05 +/− 3.1e+02
Chi2 / dof = 2.35e+03 / 2153 = 1.09
conv = FALSE
Step # 52
ts = 13.56363 h
CP = 293 to 289 mils
y = 1.96 mm
(7.92 sigma)
J = 0.132 um
Dg.pk = 1.497e−02
Dg.tr = 9.893e−03 +/− 2.58e+03
D = 5.387e−03 +/− 2.05e−06
DAi = 1.27e+05 +/− 2.4e+02
DAf = 1.243e+05 +/− 1.3e+02
Chi2 / dof = 2.7e+03 / 2446 = 1.1
conv = TRUE
Loss rate [arb. units]
Time [s]

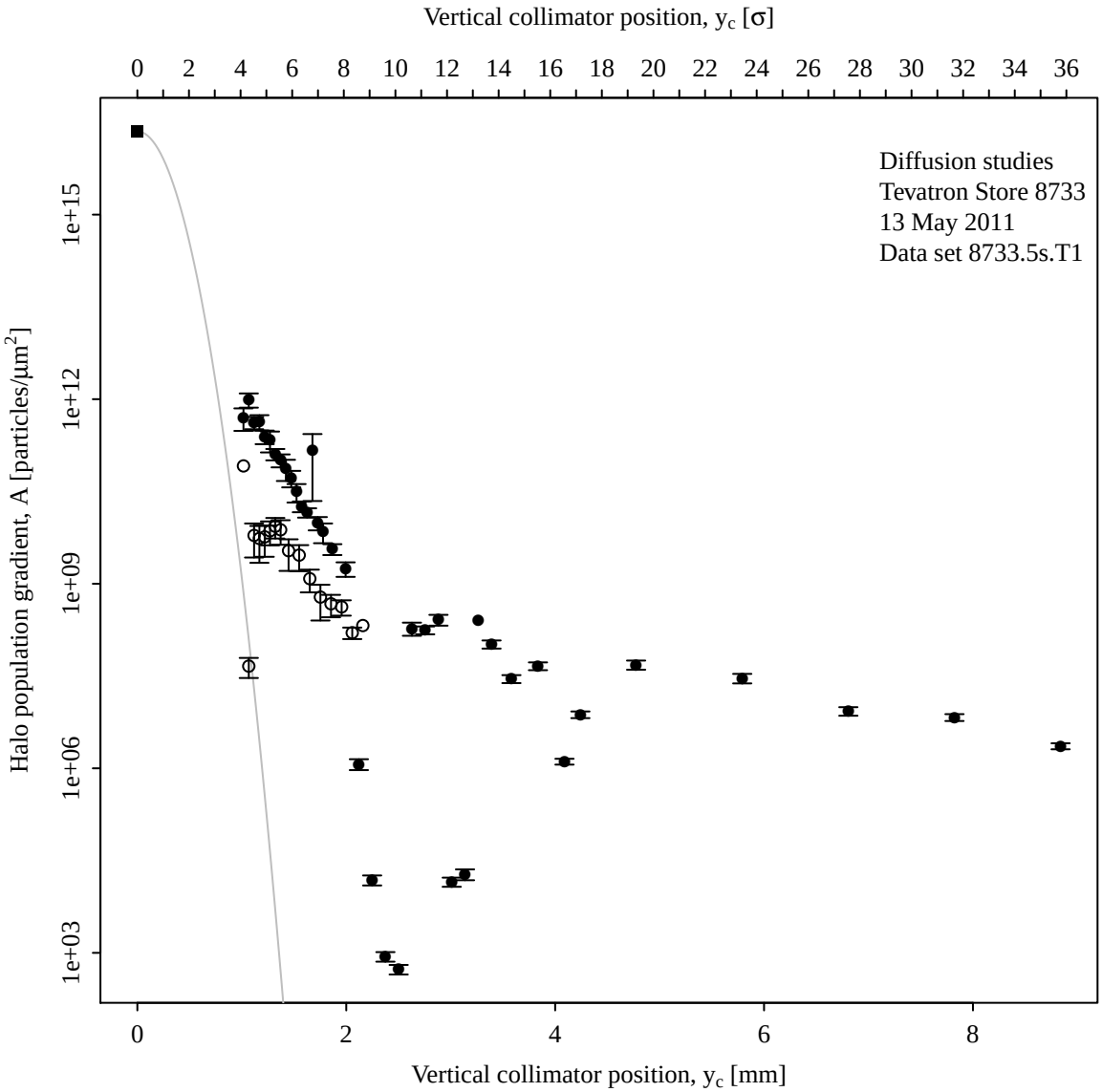

Step # 53
ts = 13.59718 h
CP = 289 to 285 mils
y = 2.06 mm
(8.33 sigma)
J = 0.146 um
Dg.pk = 1.802e−02
Dg.tr = 1.247e−01 +/− 1.47e+04
D = 1.196e−02 +/− 1.16e−06
DAi = 1.253e+05 +/− 2e+02
DAf = 1.237e+05 +/− 1.3e+02
Chi2 / dof = 2.84e+03 / 2817 = 1.01
conv = TRUE
Step # 54
ts = 13.63034 h
CP = 285 to 281 mils
y = 2.16 mm
(8.74 sigma)
J = 0.16 um
Dg.pk = 1.323e−02
Dg.tr = 1.138e−02 +/− 3.08e+03
D = 1.657e−02 +/− 3.69e−06
DAi = 1.237e+05 +/− 1.8e+02
DAf = 1.238e+05 +/− 3.4e+02
Chi2 / dof = 1.48e+03 / 1481 = 1
conv = FALSE
Loss rate [arb. units]
100000
120000
140000
160000
180000
−50
0
50
100
150
−60
−40
−20
0
20
40
Time [s]


Vertical collimator position, y_c [σ]
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36
Diffusion studies
Tevatron Store 8733
13 May 2011
Data set 8733.5s.T1
Diffusion rate, D [μm²/s]
1e+02
1e−01
1e−04
1e−07
1e−10
Fit results
Estimates from peak
Estimates from transient
Estimates from 1−par fit
0 2 4 6 8
Vertical collimator position, y_c [mm]


Vertical collimator position, y_c [σ]
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36
Diffusion studies
Tevatron Store 8733
13 May 2011
Data set 8733.5s.T1
Calibration factor, k [(loss rate units) / (particles/s)]
5e−02
5e−03
5e−04
1e−04
Kg = 0.108
0 2 4 6 8
Vertical collimator position, y_c [mm]


Vertical collimator position, y_c [σ]
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36
Steady−state loss rate, kDA [loss monitor units]
2e+05
5e+04
2e+04
5e+03
2e+03
5e+02
2e+02
Diffusion studies
Tevatron Store 8733
13 May 2011
Data set 8733.5s.T1
Initial (before step)
Final (after step)
0 2 4 6 8
Vertical collimator position, y_c [mm]


Vertical collimator position, y_c [σ]
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36
Diffusion studies
Tevatron Store 8733
13 May 2011
Data set 8733.5s.T1
Intensity decay rate:
lambda = 8.807e−06 +/− 3.26e−08 1/s
(tau = 31.54 h)
Emittance growth rates:
gamma_x = 7.711e−06 +/− 7.99e−07 1/s
gamma_y = 8.161e−06 +/− 4.86e−07 1/s
Particle flux, DA [particles/s]
1e+06
2e+06
5e+06
1e+07
2e+07
0 2 4 6 8
Vertical collimator position, y_c [mm]



Appendix B: Files and scripts

1. dataset_descriptions.txt

```
#
# Descriptions of data sets used for Tevatron diffusion analysis
#
# ID = unique dataset identification and directory name
# Sto = store number
# Da = day
# Mo = month
# Yr = year
```

```
# HH = hour
# MM = minutes
# CollName = collimator name [string]
# AxisPos = position of the beam axis in collimator coordinates [mils]
# LossMon = which loss monitor?
#   S : total losses, slow logger (TLF480 in file CLb.txt.gz)
#   T : total losses, fast logger (TLF480_15Hz, from Feb 3, 2011 at 11:40)
#   G : gated loss monitors (TLF49T1, 2, or 3, depending on the set of bunches)
# eIin = electron lens peak current [mA]
# Bu1, Bu2 = range of bunches of interest
# Notes = miscellaneous notes

  ID              Sto Da Mo   Yr HH MM CollName AxisPos  LossMon eIin Bu1 Bu2 Notes

"8467.slow"    8467 03 02 2011 11 20     F49V    -701    S 150  13  24 "test, aborted"
"8467.fast"    8467 03 02 2011 11 20     F49V    -701    T 150  13  24 "test, aborted"

"8508.nom"     8508 17 02 2011 16 00     F48V     382    T   0   1  36 "nominal"
"8508.tel"     8508 17 02 2011 17 20     F48V     382    T 140   1  36 "TEL2 on, 3.5s"

"8527.nom"     8527 25 02 2011 10 35     F48V     377    T   0   1  36 "nominal"
"8527.tel"     8527 25 02 2011 13 00     F48V     377    T 100   1  36 "TEL2 on, 3.8s"

# gated loss monitors installed March 2011

"8733.5s.TOT"  8733 13 05 2011 11 30     F48V     368   T   0   1  12 "Total losses"
"8733.5s.T1"   8733 13 05 2011 11 30     F48V     368   G   0   1  12 "Control train T1"
"8733.5s.tel"  8733 13 05 2011 11 30     F48V     368   G 400  13  24 "TEL at 5s, 3 TEL trips"
"8733.5s.T3"   8733 13 05 2011 11 30     F48V     368   G   0  25  36 "Control train T3"

"8733.4s.TOT"  8733 13 05 2011 13 43     F48V     368   T   0   1  36 "Outward, total losses"
"8733.4s.T1"   8733 13 05 2011 13 43     F48V     368   G   0   1  12 "Outward, control train T1"
"8733.4s.tel"  8733 13 05 2011 13 43     F48V     368   G 400  13  24 "TEL at 4s"
"8733.4s.T3"   8733 13 05 2011 13 43     F48V     368   G   0  25  36 "Outward, control train T3"

"8749.hi.TOT"  8749 20 05 2011 06 57     F48V     369   T   0   1  36 "total losses"
"8749.hi.T1"   8749 20 05 2011 06 57     F48V     369   G   0   7  12 "control train T1"
"8749.hi.tel"  8749 20 05 2011 06 57     F48V     369   G 872  19  24 "1/2 of train 2, 4.25s"
"8749.hi.T3"   8749 20 05 2011 06 57     F48V     369   G   0  31  36 "control train T3"
```

```
"8749.me.TOT"  8749 20 05 2011 09 05     F48V     369   T   0   1  36 "total losses"
"8749.me.T1"   8749 20 05 2011 09 05     F48V     369   G   0   7  12 "control train T1"
"8749.me.tel"  8749 20 05 2011 09 05     F48V     369   G 611  19  24 "TEL2 on 1/2 of train 2"
"8749.me.T3"   8749 20 05 2011 09 05     F48V     369   G   0  31  36 "control train T3"

"8749.lo.TOT"  8749 20 05 2011 10 00     F48V     369   T   0   1  36 "total losses"
"8749.lo.T1"   8749 20 05 2011 10 00     F48V     369   G   0   7  12 "control train T1"
"8749.lo.tel"  8749 20 05 2011 10 00     F48V     369   G 318  19  24 "TEL2 on 1/2 of train 2"
"8749.lo.T3"   8749 20 05 2011 10 00     F48V     369   G   0  31  36 "control train T3"

# pbar-only store
"8764.hin.TOT" 8764 24 05 2011 14 53     F48V     369   T   0   1   4 "pbar only, h co. in, total"
"8764.hin.nom" 8764 24 05 2011 14 53     F48V     369   G   0   1   4 "pbar only, h co. in, control"
"8764.hin.tel" 8764 24 05 2011 14 53     F48V     369   G 832  13  16 "pbar only, h co. in, 4.5s"
# T:F48VCP logged at 1 Hz starting at 17:06
# T:F48HCP moved completely out at 17:10
"8764.TOT"     8764 24 05 2011 17 58     F48V     369   T   0   1   4 "pbar only, total losses"
"8764.nom"     8764 24 05 2011 17 58     F48V     369   G   0   1   4 "pbar only, control bunches"
"8764.tel"     8764 24 05 2011 17 58     F48V     369   G 832  13  16 "pbar only, 4.5s"
# TEL2off at 19:46:07
```

2. utilities.R

```
# Matlab color palette
matlab.colors <-colorRampPalette(
  c("#00007F", "blue", "#007FFF", "cyan", "#7FFF7F", "yellow",
    "#FF7F00", "red", "#7F0000"))



# combine measurements of the same phenomenon
# with possibly different uncertainties
# return weighted mean and its error
combine <- function(x=rnorm(10), ex=runif(10, 0.5, 1.5), plot=FALSE) {
  # checks
  if(!is.vector(x) | !is.vector(ex)) stop("x and ex must be vectors")
  if(length(x)!=length(ex)) stop("x and ex must have the same length")
  if(length(ex[ex==0])) warning("Null uncertainties")
  if(length(x[is.na(x)]) | length(ex[is.na(ex)]))
    warning("Null values will be ignored")
```

```
  cut <- !is.na(x) & !is.na(ex)
  x <- x[cut]; ex <- ex[cut]
  w <- 1/ex/ex # weights
  wm <- weighted.mean(x, w=w) # weighted mean
  ewm <- 1/sqrt(sum(w)) # error on weighted mean
  if(plot) {
    dev.new()
    plot(x, 1:length(x), xlim=range(c(x-ex,x+ex)), ylim=c(0,length(x)),
         ylab="", axes=FALSE, pch=16)
    axis(1); box()
    arrows(x+ex, 1:length(x), x-ex, 1:length(x), le=0)
    points(wm,0, col="red", pch=16)
    arrows(wm+ewm, 0, wm-ewm, 0, le=0, col="red") }
  return(list(wm=wm,ewm=ewm))
}
# complex wrapper: Re=x, Im=ex
ccombine <- function(zin) {
  x <- Re(zin); ex <- Im(zin)
  m <- combine(x, ex)
  zout <- complex(re=m$wm, im=m$ewm)
  return(zout) }


# calculation of exponential decay rate of a phenomenon
# by linear interpolation of log(z)
# t = vector of times
# z = vector of corresponding observations
# assumes model y = (B-A) * exp (-lambda*(t-t0)) + A
# with negligible offset A
decay.rate <- function(t=1:20,
                       z=exp(-(1:20)/20)+rnorm(20,sd=0.02),
                       plot=FALSE) {
  if(length(t) != length(z) |
     !is.vector(t) | !is.vector(z))
    stop("'t' and 'z' must be vectors of equal length.")
  sub <- (z>0) & (!is.na(z)) & (!is.na(t))
  if(length(z[!sub]))
    warning("Warning: nonpositive and NA values will be ignored.")
  dr <- numeric(1) # decay rate
  edr <- numeric(1) # error
```

```
  x <- t[sub]; y <- log(z[sub]/max(z[sub]))
  firstpass <- TRUE # first interpolation, look for outliers
  while(firstpass) {
    N <- length(x)
    sx <- sum(x); sx2 <- sum(x*x)
    sy <- sum(y); sxy <- sum(x*y)
    D <- N*sx2-sx*sx
    A <- (sx2*sy - sx*sxy)/D
    B <- (N*sxy - sx*sy)/D
    res <- y-A-B*x
    ey2 <- sum(res*res)/(N-2)
    good <- abs(res)/mad(res) < 6 # exclude obvious outliers
    if(length(x) == length(x[good])) {
      dr <- -B
      edr <- sqrt(N*ey2/D)
      firstpass <- FALSE} else {x <- x[good]; y <- y[good]} }
  result <- list(lambda=dr, elambda=edr)
  if(plot) {dev.new(); plot(x,z);
    dev.new(); plot(x,y)
    curve(A+B*x, range(x), add=TRUE, lwd=2)
    curve(A+(B+edr)*x, range(x), add=TRUE)
    curve(A+(B-edr)*x, range(x), add=TRUE)
    points(x,y) }
  return(result)
}

# wrapper for complex usage of above routine
# Re(zin) = time; Im(zin) = function values
# Re(zout) = decay rate; Im(zout) = uncertainty
cdecay.rate <- function(zin) {
  d <- decay.rate(t=Re(zin), z=Im(zin))
  zout <- complex(real=d$lambda,imaginary=d$elambda)
  return(zout)}

# running decay rate over window of width nw
decay.rate.w <- function(t=1:100, z=exp(-(1:100)/100)+rnorm(100,sd=0.02),
                         nw=round(length(t)*0.2)+1, plot=FALSE) {
  dr <- numeric(length(t)) # running decay rates
  edr <- numeric(length(t)) # errors
  for(i in 1:length(t)){
```

```
    j1 <- i-round(nw/2); j2 <- i+round(nw/2)
    if(j1 < 1) {j1 <- 1; j2 <- min(1+nw,length(t))}
    if(j2 > length(t)) {j1 <- max(1,length(t)-nw); j2 <- length(t)}
    x <- t[j1:j2]; y <- z[j1:j2]
    rate <- decay.rate(x,y)
    dr[i] <- rate$lambda
    edr[i] <- rate$elambda }
  result <- cbind(dr, edr)
  if(plot) {dev.new()
            plot(t, dr, type="l", ylim=c(min(dr-edr),max(dr+edr)), lwd=2)
            points(t, dr+edr, type="l")
            points(t, dr-edr, type="l") }
  return(result)
}

# running decay rate by smoothing and analytical spline derivative
decay.rate.s <- function(t=1:100, z=exp(-(1:100)/100)+rnorm(100,sd=0.02),
                         plot=TRUE) {
  y <- z; ly <- log(z)
  ysp <- smooth.spline(t,y)
  lysp <- smooth.spline(t,ly)
  lambda <- -predict(object=lysp, x=t, deriv=1)$y
  if(plot) {dev.new()
    plot(t,z, ylab="Data + smoothed function", las=1, xlim=range(t))
    points(t, ysp$y, type="l", col="gray", lwd=2)
    plot.window(xlim=range(t), ylim=range(lambda), las=1)
    axis(4); mtext("Decay rate", side=4)
    points(t,lambda,type="l")}
  return(lambda)
}



# TEL2 peak current vs. T:L2MCV1 modulator voltage
# with anode bias at -200 V
# trigger mode 2, pulsing on second train
Itrain <- splinefun(x=c(seq(0,550,by=50),590),
                    y=c(0, 12.7, 52.4, 100.1, 154.1, 216.1, 276.5, 325.7, 379.7, 436.9, 497.3, 552.8, 596.0))
```

```
# Read files formatted by Batch Data Logger Extractor
ReadACNETData <- function(fn="CFBI_NG.txt.gz") {
cat("Reading file ", fn, "...\n")
dn <- gsub(".txt.gz", "", fn) # name of data frame
t0 <- scan(fn, what="character", skip=1, nlines=1) # get initial time
t0 <- t0[6]
h <- as.numeric(substr(t0,1,2))
m <- as.numeric(substr(t0,4,5))
s <- as.numeric(substr(t0,7,8))
assign("T0", h+m/60+s/3600, envir=.GlobalEnv)
cat("Start time: ", T0, " h\n")
cnam <- scan(fn, what="character", skip=2, nlines=1, sep="\t") # column names
cnam <- substr(cnam, 3, 12) # strip logger names
cnam <- gsub("[^[:alnum:]]", "", cnam) # remove non alphanumeric characters
if(length(cnam)<72) cnam <- gsub("0", "", cnam) # remove 0s from scalars
cnam[seq(1,length(cnam),by=2)] <-
  paste("t", cnam[seq(2,length(cnam),by=2)], sep="") # times
assign(dn, read.table(fn, header=FALSE, skip=3, colClasses="numeric",
                      na.strings=c("NA","999999"), col.names=cnam),
       envir= .GlobalEnv) # fill data frame
}



# hourglass factor for luminosity
hg <- function(s1x=60e-6, # everything in meters
               s2x=40e-6,
               s1y=66e-6,
               s2y=44e-6,
               s1z=0.6,
               s2z=0.6,
               b1x=0.5,
               b2x=0.5,
               b1y=0.6,
               b2y=0.6) {
  ttx <- 2 * (s1x^2+s2x^2) / (s1z^2+s2z^2) / ( (s1x/b1x)^2 + (s2x/b2x)^2 )
  tty <- 2 * (s1y^2+s2y^2) / (s1z^2+s2z^2) / ( (s1y/b1y)^2 + (s2y/b2y)^2 )
  tx <- sqrt(ttx)
  ty <- sqrt(tty)
  f <- function(t) {exp(-t*t) / sqrt( (1+t*t/ttx) * (1+t*t/tty) * pi )}
```

```
  I <- integrate(f, -Inf, Inf)
  h <- I$value
  dh <- I$abs.error
  return(list(h=h, dh=dh))
}
```

3. diana.R

```
### R code from vignette source 'diana.Rnw'

Initialize <- function() {

### CONTROL PARAMETERS ###

# perform data collection
# 0 = fresh data collection on all defined data sets
# numeric vector = fresh data collection on a subset
# string = use existing data file
#DATA.SOURCE <<- 0
DATA.SOURCE <<- "TevDiffusion2011.RData"

# perform data analysis
# 0 = fresh data analysis on all defined data sets
# numeric vector = fresh data analysis on a subset
# string = use existing results file
#RESULTS.SOURCE <<- 0
RESULTS.SOURCE <<- "Results_v8.RData"
STEPS.SUBSET <<- 1
DEBUG.MODE <<- FALSE
BOOTSTRAP <<- FALSE # estimate parameter errors with bootstrapping technique

#message("\n\nADD LIFETIME AND EMITTANCE GROWTH DIAGNOSTICS!\n\n")
######


cornicina <<- paste(rep("=", 60), collapse="")
message("\n\n", cornicina, "\n",
    "  Diffusion analysis of collimator scans\n",
    cornicina, "\n", date(), "\n\n")
message("Initializing...\n")
```

```
library(MASS); library(nlme); library(xtable)
options(continue="  ")

ddir <<- getwd() # default directory
on.exit({setwd(ddir)}) # cleanup on exit

# seed for random number generation
set.seed(12345)

# useful constants
um.mil <<- 25.4 # microns per mil
std.lvl <<- pnorm(1)-pnorm(-1) # standard confidence level

source("../utilities.R") # define utility functions
idem <<- function(x){x}
# beam size calculation
sigma2 <<- function(sigma1=0.5, dpp=1.5e-4,
               beta2=100, D2=1000,
               beta1=50, D1=600) {
    sigma2 <- sqrt(beta2/beta1*(sigma1^2-(D1*dpp)^2)+(D2*dpp)^2)
    return(as.numeric(sigma2))}
# time in specified range
t.in.range <<- function(t=seq(-1,6,by=0.2), # time variable
                       t1=1:4, # vector of lower limits
                       t2=1.5:4.5){ # vector of upper limits
  if(length(t1) != length(t2)) stop("t1 and t2 lengths differ")
  tin <- logical(length(t))
  for(i in 1:length(t1)){ # check if t is within any range
    tin <- tin | (t1[i]<=t & t<=t2[i]) }
  return(tin) } # return logical
# utility function to explore collimator steps
ExploreSteps <<- function(dir="../Store8733_110513", t1=10, t2=14,
                          coll="F48", loss="G2"){
  curdir <- getwd()
  setwd(dir)
  # fast logging of collimator positions?
  FASTCPLOG <- FALSE
  if(length(grep("Store8764", dir))==1 & t1>(17.1)) FASTCPLOG <- TRUE
  if(FASTCPLOG) flist <- c("TF48CP_1Hz.txt.gz") else flist <- c("CP.txt.gz")
  if(loss=="S") flist <- c(flist, "CLb.txt.gz")
```

```
  if(loss=="T") flist <- c(flist, "TLF480_15Hz.txt.gz")
  if(substring(loss,1,1)=="G") flist <- c(flist, "TLF49T_15Hz.txt.gz")
  for(fn in flist) ReadACNETData(fn)
  if(FASTCPLOG){
      tc <<- TF48CP_1Hz$tF48VCP/3600+T0
      cp <<- TF48CP_1Hz$F48VCP
  } else {
      tc <<- CP$tF48VCP/3600+T0
      cp <<- CP$F48VCP}
  if(coll=="F49"){
    tc <<- CP$tF49VCP/3600+T0
    cp <<- CP$F49VCP}
  if(loss=="S"){
    tl <<- CLb$tLF48/3600+T0
    lo <<- CLb$LF48}
  if(loss=="T"){
    tl <<- TLF480_15Hz$tLF48/3600+T0
    lo <<- TLF480_15Hz$LF48}
  if(loss=="G1"){
    tl <<- TLF49T_15Hz$tLF49T1/3600+T0
    lo <<- TLF49T_15Hz$LF49T1}
  if(loss=="G2"){
    tl <<- TLF49T_15Hz$tLF49T2/3600+T0
    lo <<- TLF49T_15Hz$LF49T2}
  if(loss=="G3"){
    tl <<- TLF49T_15Hz$tLF49T3/3600+T0
    lo <<- TLF49T_15Hz$LF49T3}
  cut <- t1<=tl & tl<=t2
  x <- tl[cut]; y <- lo[cut]
  dev.new(); plot(x, y, xlim=c(t1, t2), type="l")
  cut <- t1<=tc & tc<=t2
  x <- tc[cut]; y <- cp[cut]
  plot.window(xlim=c(t1, t2), ylim=range(y, finite=TRUE))
  points(x, y, type="o", col="red")
  axis(4)
  setwd(curdir)
}
# utility to select regions of interest for interpolation
SelectRegions <<- function(t=seq(10, 14, by=0.1), # fake time and
  y=t, # loss data
```

```
  guesses=11:13, # initial guesses [h]
  largew=4/60, # wide window to select initial, step, and final times
  smallw=2/3600){ # narrow window to determine step time precisely
  guess1 <- guesses
  guess2 <- guesses
  for(i in seq(along=guesses)){
    for( window in c(largew, smallw) ){ # wide and narrow time windows
      sub <- abs(t-guesses[i]) <= window
      plot(t[sub], y[sub])
      if(window==largew){
        guess1[i] <- locator(1)$x
        guesses[i] <- locator(1)$x
        guess2[i] <- locator(1)$x
      } else { guesses[i] <- locator(1)$x } } }
  GU <- cbind(guess1, guesses, guess2)
  colnames(GU) <- c("t1", "ts", "t2")
  write.table(GU, file="tmp.txt", row.names=FALSE)
  return(GU)
}

# utility function to determine largest time interval for which a
# certain condition is true (useful for calibration of losses)
FindInterval <<- function(t=0:100, x=sin(sqrt(t)), co=x>0){
  if(length(x)!=length(t) | length(co)!=length(t))
    stop("Lengths of vectors differ.")
#  if(!all(t==sort(t, na.last=TRUE))) warning("Time vector is not sorted.")
  subset <- !is.na(t) & !is.na(x) & !is.na(co)
  t <- t[subset]; x <- x[subset]; co <- co[subset]
  TI <- list(ti=min(t, na.rm=TRUE), tf=max(t, na.rm=TRUE))
  n <- length(t)
  if(n<=2) {warning("Vectors are short!"); return(TI)}
  le <- 1; ki <- 1; kf <- 2
  for(i1 in 1:(n-1)){ # brute-force algorithm
    for(i2 in (i1+1):n){
      if(all(co[i1:i2], na.rm=TRUE) & (i2-i1+1)>le) {
        ki <- i1; kf <- i2; le <- (i2-i1+1) }
  }}
  TI$ti <- t[ki]; TI$tf <- t[kf]
  return(TI)}
```

```
# utility function to define model weights
# according to local variance of response
LocalRMS <<- function(x=rnorm(100), n=20){
  l <- length(x)
  s0 <- mad(x, na.rm=TRUE)
  lrms <- rep(s0, l)
  if(n > l) {
    warning("Length of neighborhood is larger than vector length.")
    return(lrms) }
  j <- seq(1, l)
  i1 <- pmax(1, round(j - n/2))
  i2 <- pmin(l, round(j + n/2))
  edge1 <- i1==1
  edge2 <- i2==l
  i2[edge1] <- n
  i1[edge2] <- l-n+1
  for(j in 1:l){
    xx <- x[i1[j]:i2[j]]
    s <- mad(xx, na.rm=TRUE)
    if(1e-1*s0<s & s<1e1*s0) lrms[j] <- s }
  return(lrms)}

options(
    SweaveHooks=list(fig=function() { # common Sweave plot options
        wi<-6; he<-6; pt<-10; fo<-"Times"
        ps.options(width=wi, height=he, pointsize=pt, family=fo, bg="white")
        pdf.options(width=wi, height=he, pointsize=pt, family=fo, bg="white")
        par(oma=rep(0,4), mar=c(4.2,4.2,0.2,0.2)) }))

# utility to compare data sets
CompareSets <<- function(set.list=28:29, co=c("darkblue", "cyan"),
                         jwhich=list(19:39, 19:39),
                         xvar="xs",
                         xlim.par=NA,
                         core=FALSE,
                         elens=FALSE,
                         plot.id="xyz"){
  if(!exists("Res")) stop("Results not defined.")
  nsets <- length(set.list)
  x <- xs <- J <- y <- ylo <- yup <- sym <- list(nsets)
```

```
  xl <- xsl <- Jl <- yl <- c(Inf, -Inf)
  for(i in seq(along=set.list)){
    k <- set.list[i]
    j <- 1:nrow(Diff$StepTimes[[k]])
    if(!is.na(jwhich)) j <- jwhich[[i]]
    if(all(is.na(Res$D[[k]][j]))) stop("Not all sets have results.")
    sym[[i]] <- 1+15*Res$stepin[[k]][j]
    x[[i]] <- abs(0.5*(Res$xci[[k]][j]+Res$xcf[[k]][j])/1e3) # [mm]
    xs[[i]] <- x[[i]] / Diff$syC[k] # [sigma]
    J[[i]] <- abs(0.5*(Res$Jci[[k]][j]+Res$Jcf[[k]][j]))
    y[[i]] <- Res$D[[k]][j]
    ylo[[i]] <- Res$D[[k]][j] - Res$eD[[k]][j]
    yup[[i]] <- Res$D[[k]][j] + Res$eD[[k]][j]
    if(core){
      sym[[i]] <- c(0, 15, sym[[i]])
      x[[i]] <- c(0, 0, x[[i]])
      xs[[i]] <- c(0, 0, xs[[i]])
      J[[i]] <- c(0, 0, J[[i]])
      y[[i]] <- c(Res$D0x[k], Res$D0y[k], y[[i]])
      ylo[[i]] <- c(Res$D0x[k]-Res$eD0x[k], Res$D0y[k]-Res$eD0y[k], ylo[[i]])
      yup[[i]] <- c(Res$D0x[k]+Res$eD0x[k], Res$D0y[k]+Res$eD0y[k], yup[[i]])
    }
    xl[1] <- min(xl[1], min(x[[i]]))
    xl[2] <- max(xl[2], max(x[[i]]))
    xsl[1] <- min(xsl[1], min(xs[[i]]))
    xsl[2] <- max(xsl[2], max(xs[[i]]))
    Jl[1] <- min(Jl[1], min(J[[i]]))
    Jl[2] <- max(Jl[2], max(J[[i]]))
    yl[1] <- min(yl[1], min(y[[i]]))
    yl[2] <- max(yl[2], max(y[[i]]))
  }
#  rndi <- ceiling(runif(5, min=0, max=26))
#  rndtxt <- paste(letters[rndi], sep="", collapse="")
  rndtxt <- plot.id
  basename <- paste(Diff$ddir, "/", "tmp_comparison_", rndtxt, sep="")
  pdf(file=paste(basename, ".pdf", sep=""))
  dev.control(displaylist="enable") # enable recording
  par(oma=rep(0,4), mar=c(3.2,3.2,3.2,0.2), mgp=c(2,0.5,0), tcl=-0.25)
  if(!is.finite(xl[1])) xl[1] <- 0
  if(!is.finite(xl[2])) xl[2] <- 10
```

```
  if(!is.finite(xsl[1])) xsl[1] <- 0
  if(!is.finite(xsl[2])) xsl[2] <- 10
  if(!is.finite(Jl[1])) Jl[1] <- 0
  if(!is.finite(Jl[2])) Jl[2] <- 1
  if(!is.finite(yl[1])) yl[1] <- 1e-12
  if(!is.finite(yl[2])) yl[2] <- 1e2
  if(xvar=="x") {
      xx <- x
      xlim <- xl
      xlabel <- expression(paste("Vertical collimator position, ", y[c], " [mm]", sep="")) }
  if(xvar=="xs") {
      xx <- xs
      xlim <- xsl
      xlabel <- expression(paste("Vertical collimator position, ", y[c], " [", sigma, "]", sep="")) }
  if(xvar=="J") {
      xx <- J
      xlim <- Jl
      xlabel <- expression(paste("Vertical collimator action, ", J[c], " [", mu, "m]", sep="")) }
  if(!is.na(xlim.par)) xlim <- xlim.par
  plot(c(1,2), c(1,2), xlim=xlim, ylim=yl, log="y", type="n",
    xlab=xlabel,
    ylab=expression(paste("Diffusion rate, D [", mu*m^2/s, "]", sep="")))
  for(i in 1:nsets){
    k <- set.list[i]
    if(elens & Diff$eIin[k]>0){
      elens.r <- c(Diff$ris[k], Diff$ros[k])
      abline(v=elens.r, col=co[i])}
    points(xx[[i]], y[[i]], pch=sym[[i]], col=co[i])
    arrows(xx[[i]], ylo[[i]], xx[[i]], yup[[i]], le=0.05, angle=90, code=3,
           col=co[i])
  }
  legend("bottomright", bty="n", text.col=co, legend=Diff$ID[set.list])
  dev.copy2eps(file=paste(basename, ".eps", sep=""))
  dev.off()
  IncludePlotInLatex(file=basename, flo=FALSE,
                     cap=paste("Comparison of data sets", Diff$ID[set.list]),
                     lab=paste("fig:comparison", rndtxt, sep="_"))
}

# to include a figure in LaTeX, as floating object or simple graphics
```

```
IncludePlotInLatex <<- function(file="Rplots",
                                flo=TRUE,
                                cap="This is the caption.",
                                lab="fig:dummy_label"){
if(flo){
    fi <- paste("\n\\begin{figure}\n\\includegraphics{", file,
            "}\n\\caption{", cap,
            " \\label{", lab, "}}\n\\end{figure}\n", sep="")
} else {
    fi <- paste("\n\\begin{center}\n\\includegraphics{", file,
                "}\n\\end{center}\n", sep="")
}
cat(fi)
}

}

Initialize()

Cleanup <- function(){
  setwd(ddir)
  for(i in 1:3) gc()
  message("\n\n", cornicina, "\n",
    "  End of diffusion analysis.\n",
    cornicina, "\n", date(), "\n\n")
}



DefineModel <- function() {
# collimator position vs. time, linear
Jc <<- function(t, Jci=0.04, Jcf=0.05, dt=1){
  Jc <- 0*t + Jci # initial value
  subset <- dt<=t
  Jc[subset] <- Jcf # final value
  subset <- 0<t & t<dt
  Jc[subset] <- Jci + (Jcf-Jci)*t[subset]/dt # collimator moving
  return(Jc)
}

# slope of distribution vs. time, linear
```

```
Ac <<- function(t, Ai=1, Af=0.9, dt=1){
  Ac <- 0*t + Ai # initial value
  subset <- dt<=t
  Ac[subset] <- Af # final value
  subset <- 0<t & t<dt
  Ac[subset] <- Ai + (Af-Ai)*t[subset]/dt # collimator moving
  return(Ac)
}

# initial, final, and asymptotic distributions, linear
flin <<- function(J, A=1, J0=0.04){
  f <- 0*J # default outside range
  J0 <- abs(J0) # define edge as positive
  subset <- abs(J)<J0 # triangular with cusp at J=0
  f[subset] <- A*(J0 - abs(J[subset])) # linear
  return(f) }

# evolution of the distribution function (der=0)
# or its derivative (der=1), analytical formulas
fs <<- function(J, t, Ai=1, Af=0.9, Jci=0.04, Jcf=0.05, D=1e-5, dt=1, der=0) {
  if(der!=0 & der!=1) stop("Derivative ('der') must be 0 or 1.")
  fs <- 0 # default outside collimators
  if(Jci>Jcf) IN <- TRUE else IN <- FALSE # inward or outward step?
  if(t<0){ # initial distribution
    if(der==0) fs <- flin(J, A=Ai, J0=Jci)
    if(der==1) fs <- -Ai
  } else{ # evolution
    Jco <- Jc(t, Jci=Jci, Jcf=Jcf, dt=dt) # collimator position
    Aco <- Ac(t, Ai=Ai, Af=Af, dt=dt) # slope of asymptotic solution
    if(0<=J & J<=Jco){ # inside collimators
      s <- sqrt(2*D*t) # Gaussian sigma
      # terms common to both inward and outward step
      if(der==0) fs.base <- -pnorm(-J/s) * (Ai*(Jci-J) - Aco*(Jco-J)) +
          pnorm((J-2*Jco)/s) * (Ai*(Jci+J-2*Jco) - Aco*(J-Jco)) +
          s/sqrt(2*pi) * (
            exp(-0.5*(J/s)^2) * (Aco-Ai) +
            (-1)*exp(-0.5*((2*Jco-J)/s)^2) * (Aco-Ai) )
      if(der==1) fs.base <- (Ai-Aco)*(pnorm(-J/s) + pnorm((J-2*Jco)/s)) +
          (Ai*Jci-Aco*Jco)/sqrt(2*pi)/s *
              (exp(-0.5*(J/s)^2)+exp(-0.5*((J-2*Jco)/s)^2))
```

```
      # add specific terms
      if(IN) { # inward step
        if(der==0) fs <- fs.base +
            (-1)*Ai*(Jci+J-2*Jco) +
            2 * pnorm((Jco-J)/s) * Ai * (Jci-Jco)
        if(der==1) fs <- fs.base +
            (-1)*Ai + 2*Ai*(Jco-Jci)*exp(-0.5*((Jco-J)/s)^2)/sqrt(2*pi)/s
      } else { # outward step
        if(der==0) fs <- fs.base +
          pnorm((Jci-J)/s) * Ai * (Jci-J) +
          (-1)*pnorm((Jci+J-2*Jco)/s) * Ai * (Jci+J-2*Jco) +
          s/sqrt(2*pi) * (
            exp(-0.5*((Jci-J)/s)^2) * Ai +
            (-1)*exp(-0.5*((Jci+J-2*Jco)/s)^2) * Ai )
        if(der==1) fs <- fs.base +
            (-1)*Ai*(pnorm((Jci-J)/s)+pnorm((Jci+J-2*Jco)/s))
      }
    }
  }
  return(fs)
}
fv <<- Vectorize(fs, vectorize.args=c("J","t")) # vectorize the function

# measured loss rate
# extension of Mess and Seidel, NIM A 351, 279 (1994)
# G. Stancari, arXiv:1108.5010 [physics.acc-ph]
#
# t - t0 = time relative to initial collimator movement (s)
# D = diffusion rate (um^2/s)
# K = loss monitor conversion factor (V / (particles/s))
# B = loss monitor background (V)
# Ai = initial slope of particle distribution
# Af = final slope of particle distribution
# Jci = initial collimator action (um)
# Jcf = final collimator action (um)
# dt = time to move collimators from Jci to Jcf (s)

# loss rate vs. time
# particle flux at the collimator L = -D * (df/dJ)|J=Jc
# with conversion constant k and background B
```

```
loss.rate <<- function(t, t0=0, Ai=1, Af=0.9, Jci=0.04, Jcf=0.05, D=1e-5, dt=1,
                       K=1, B=0) {
  t <- t-t0; D <- abs(D)
#  dt <- abs(dt)
#  Ai <- abs(Ai); Af <- abs(Af); Jci <- abs(Jci); Jcf <- abs(Jcf)
#  K <- abs(K)
  L <- -Ai + 0*t # default before step
  subset <- t>0 # evolution
  s <- sqrt(2*D*t[subset]) # Gaussian sigmas
  Jco <- Jc(t[subset], Jci=Jci, Jcf=Jcf, dt=dt) # collimator positions
  Aco <- Ac(t[subset], Ai=Ai, Af=Af, dt=dt) # slopes
  if(Jci>Jcf) IN <- TRUE else IN <- FALSE # inward or outward?
  # common terms
  L[subset] <- 2*(Ai-Aco)*pnorm(-Jco/s) +
    2/sqrt(2*pi)/s * (Ai*Jci-Aco*Jco) * exp(-0.5*(Jco/s)^2)
  if(IN) {
      L[subset] <- L[subset] - Ai - 2/sqrt(2*pi)/s * Ai * (Jci-Jco)
  } else {
      L[subset] <- L[subset] - 2 * Ai * pnorm((Jci-Jco)/s) }
  return(-K*D*L+B)
}

}
DefineModel()



getOption("SweaveHooks")[["fig"]]()
# sample parameters
J1 <- 0.040; J2 <- 0.050 # initial and final collimator actions [um]
dt <- 1 # time to move collimator from Jci to Jcf [s]
A1 <- 1 # initial slope of distribution function [um^-2]
A2 <- A1*(J1/J2) # final slope
D <- 1e-5 # diffusion coefficient [um^2/s]
# choose times to plot
tscale.short <- abs(J1-J2)^2/D
tscale.long <- min(J1,J2)^2/D
tt <- c(seq(0, dt, length=5), # collimator moving
        c(3*dt, tscale.short, 2*tscale.short), # short term after
        c(5*tscale.short, tscale.long, 2*tscale.long)) # long term after
co <- gray(seq(0.6, 0, length=length(tt))) # define colors, grayscale
```

```
Jmin <- 0; Jmax <- max(J1,J2) # plot range
layout(matrix(1:2, nrow=1, ncol=2)) # plot layout
for(i in 1:2) {
  if(i==1){ # left plot, inward
    JJci<-J2; JJcf<-J1; Ai<-A2; Af<-A1}
  if(i==2){ # right plot, outward
    JJci<-J1; JJcf<-J2; Ai<-A1; Af<-A2}
  JJ <- Jc(tt, Jci=JJci, Jcf=JJcf, dt=dt) # calculate coll. pos.
  AA <- Ac(tt, Ai=Ai, Af=Af, dt=dt) # calculate slopes
  ylabel <- expression(paste("Distribution function [", mu*m^{-1}, "]", sep=""))
  if(i==2) ylabel <- ""
  plot(c(Jmin,Jmax),
     c(0, max(JJci*Ai,JJcf*Af)),
     type="n",
     xlab=expression(paste("Action [", mu, "m]", sep="")),
     ylab=ylabel)
abline(v=JJ, col=co, lwd=2) # plot collimator positions
for(i in seq(along=tt)){ # plot asymptotic functions
  curve(fv(J=x, t=tt[i], Ai=Ai, Af=Af, Jci=JJci, Jcf=JJcf, D=D, dt=dt),
        Jmin, Jmax, add=TRUE, col=co[i], lwd=2) }
legend("bottomleft", bty="n", # describe times for each curve
       title="Time [s]", title.col="black",
       legend=formatC(tt, drop0=TRUE), col=co, text.col=co, cex=0.8)
}


getOption("SweaveHooks")[["fig"]]()
# sample parameters
J1 <- 0.040; J2 <- 0.050 # initial and final collimator actions [um]
dt <- 1 # time to move collimator from Jci to Jcf [s]
D <- c(1e-6, 1e-5, 1e-4) # show the effect of different diffusion coeff.
K <- 1; B <- 0
tscale.short <- abs(J1-J2)^2/min(D) # estimate time scales
tscale.long <- min(J1,J2)^2/min(D)
tmax <- tscale.short/2
co <- gray(seq(0.5, 0, length=length(D))) # define colors, grayscale
Jmin <- 0; Jmax <- max(J1,J2) # plot range
layout(matrix(1:2, nrow=1, ncol=2)) # plot layout
for(i in 1:2) {
  if(i==1){ # left plot, inward
```

```
    Jci<-J2; Jcf<-J1
    ylabel <- "Loss rate [arb. units]"
    legend.pos <- "topright"}
  if(i==2){ # right plot, outward
    Jci<-J1; Jcf<-J2
    ylabel <- ""
    legend.pos <- "bottomright"}
  ADD <- FALSE
  for(j in 1:length(D)){
    Ai <- 1/D[j] # scale slope to keep steady-state rates constant
    Af <- Ai*Jci/Jcf
    if(j>1) ADD <- TRUE
    curve(loss.rate(t=x, Jci=Jci, Jcf=Jcf, Ai=Ai, Af=Af, D=D[j],
                    dt=dt, K=K, B=B), -0.1*tmax, tmax,
          xlab="Time [s]", ylab=ylabel, add=ADD, lwd=2, col=co[j])}
  if(i==1) legend(legend.pos, bty="n", # print diffusion rate for each curve
     title=expression(paste("Diffusion rate [", mu*m^2/s, "]")),
     title.col="black",
     legend=formatC(D, format="E", di=1), col=co, text.col=co)
}


# Collect the data relevant for the diffusion analysis
# Inputs: data set description file
#         lattice functions
# Outputs: data object (return value and R data file)
CollectData <- function(default.dir=getwd(),
                        datasets.fn="dataset_descriptions.txt",
                        lattice.fn="lattice_functions.txt",
                        steps.fn="collimator_steps.txt",
                        output.fn="tmp_TevDiffusion2011.RData",
                        datasubset=0) {
message("Collecting data...\n")
setwd(default.dir)

# Read dataset descriptions
DS <- read.table(datasets.fn, header=TRUE, stringsAsFactors=FALSE)
nsets <- nrow(DS)
if(datasubset==0) datasubset <- 1:nsets
```

```
# Read lattice functions
LAT <- read.table(lattice.fn)

# Prepare global thermonuclear data container
Diff <- list(
# useful general info
  ddir = default.dir,
  ACNETdir=paste("../Store", # name of directory with ACNET data
    DS[, "Sto"], "_", DS[, "Yr"]-2000,
    formatC(DS[, "Mo"], width=2, flag="0"),
    formatC(DS[, "Da"], width=2, flag="0"), "/", sep=""),
  ACNETfiles=list(),
  ACNETt0=list(),
  rubber.stamp=paste("Diffusion studies\nTevatron Store ", # plot stamp
    DS[, "Sto"], "\n",
    DS[, "Da"], " ",
    month.abb[DS[, "Mo"]], " ",
    DS[, "Yr"], "\n",
    "Data set ", DS[, "ID"], sep=""),
  ti = DS[, "HH"] + DS[, "MM"]/60, # approximate initial time
  tf = DS[, "HH"] + DS[, "MM"]/60 + 10/60,
  LAT = LAT, # lattice data frame
# electron lens
  rig = 4.5, # inner radius at gun [mm]
  rog = 7.62, # outer radius at gun [mm]
  tSG = list(),
  SG = list(),
  Bg = numeric(nsets),
  tSM = list(),
  SM = list(),
  Bm = numeric(nsets),
  teV = list(),
  eV = list(),
  eVm = numeric(nsets),
  teI = list(),
  eI = list(),
  eIm = numeric(nsets),
  ri=numeric(nsets),
  ris=numeric(nsets),
  ro=numeric(nsets),
```

```
  ros=numeric(nsets),
# beam
  tN = list(),
  N = list(),
  Nm = numeric(nsets),
  tdpp = list(),
  dpp = list(),
  dppm = numeric(nsets),
  tSL = list(),
  sxSL = list(),
  sySL = list(),
  sxSLm = numeric(nsets),
  sySLm = numeric(nsets),
  sxC = numeric(nsets),
  syC = numeric(nsets),
  sxTEL2 = numeric(nsets),
  syTEL2 = numeric(nsets),
# collimator
  StepTimes = list(),
  tCP = list(),
  CP = list(),
# losses
  tLo = list(),
  Lo = list())

# include information contained in data set descriptions
Diff <- c(as.list(DS), Diff)

for( i in datasubset ) {

setID <- Diff$ID[i]
message("Collecting data for set ", setID, "...\n")

# collimator experiment times: initial, step, final [h]
cstep.file <- paste(default.dir, "/", setID, "/", steps.fn, sep="")
CS <- read.table(cstep.file, header=TRUE)
# save some data before and after experiments
Diff$ti[i] <- ti <- min(CS, na.rm=TRUE)-5/60
Diff$tf[i] <- tf <- max(CS, na.rm=TRUE)+5/60
Diff$StepTimes[[i]] <- as.matrix(CS)
```

```
# switch to ACNET data directory and read data
setwd(Diff$ACNETdir[i])
# beam and e-lens files
flist <- c("SBDI.txt.gz", "SL.txt.gz", "SBDM.txt.gz",
           "ELa.txt.gz", "ELb.txt.gz", "ELc.txt.gz")
# fast logging of collimator positions?
if( Diff$Sto[i]==8764 & ti>17.021 )
  flist <- c(flist, "TF48CP_1Hz.txt.gz") else flist <- c(flist, "CP.txt.gz")
# which loss monitor?
if( DS[i, "LossMon"]=="S") flist <- c(flist, "CLb.txt.gz")
if( DS[i, "LossMon"]=="T") flist <- c(flist, "TLF480_15Hz.txt.gz")
if( DS[i, "LossMon"]=="G") flist <- c(flist, "TLF49T_15Hz.txt.gz")

T0v <- numeric(length(flist)) # start times (almost always equal)
names(T0v) <- c("SBDI", "SL", "SBDM", "ELa", "ELb", "ELc", "CP", "LO")
if(length(T0v) != length(flist))
  warning("Length of T0v is different from length of flist.")
for(ii in seq(along=flist)) {ReadACNETData(flist[ii]); T0v[ii] <- T0}
if( sum(abs(diff(T0v))) !=0 )
  message("Not all T0's are equal for data set ", Diff$ID[i])
Diff$ACNETfiles[[i]] <- flist
Diff$ACNETt0[[i]] <- T0v

# Retrieve e-lens parameters
tSG <- ELa$tL2SGT / 3600 + T0v["ELa"]   # gun solenoid [kG]
SG <- ELa$L2SGT
sub <- t.in.range(tSG, CS[, "t1"], CS[, "t2"]) & SG>0.5 & !is.na(SG)
Diff$Bg[i] <- Bg <- mean(SG[sub])
sub <- ti<=tSG & tSG<=tf
Diff$tSG[[i]] <- tSG[sub]
Diff$SG[[i]] <- SG[sub]

tSM <- ELa$tL2SMT / 3600 + T0v["ELa"]
SM <- ELa$L2SMT  # main solenoid [kG]
sub <- t.in.range(tSM, CS[, "t1"], CS[, "t2"]) & SM>0.5 & !is.na(SM)
Diff$Bm[i] <- Bm <- mean(SM[sub])
sub <- ti<=tSM & tSM<=tf
Diff$tSM[[i]] <- tSM[sub]
Diff$SM[[i]] <- SM[sub]
```

```
teV <- ELa$tL2MCV1 / 3600 + T0v["ELa"]
eV <- ELa$L2MCV1   # modulator voltage setting [kV]
sub <- t.in.range(teV, CS[, "t1"], CS[, "t2"]) & eV>10 & !is.na(eV)
Diff$eVm[i] <- eVm <- mean(eV[sub]) * 8 / 1e3
sub <- ti<=teV & teV<=tf
Diff$teV[[i]] <- teV[sub]
Diff$eV[[i]] <- eV[sub]

teI <- ELc$tL2COLI / 3600 + T0v["ELc"]
eI <- ELc$L2COLI
sub <- t.in.range(teI, CS[, "t1"], CS[, "t2"]) & eI>0 & !is.na(eI)
Diff$eIm[i] <- eIm <- mean(eI[sub]) # average current [mA]
sub <- ti<=teI & teI<=tf
Diff$teI[[i]] <- teI[sub]
Diff$eI[[i]] <- eI[sub]

# inner and outer e-beam radii [mm]
Diff$ri[i] <- Diff$rig*sqrt(Bg/Bm)
Diff$ro[i] <- Diff$rog*sqrt(Bg/Bm)



# Retrieve beam parameters

# intensity
# select affected bunches
tabu <- paste("tSBDAIS",
              as.character(DS[i, "Bu1"]:DS[i, "Bu2"]), sep="")
abu <- paste("SBDAIS", as.character(DS[i, "Bu1"]:DS[i, "Bu2"]), sep="")
# measurement times
tI <- rowMeans( SBDI[,tabu] ) / 3600 + T0v["SBDI"]
# intensity of bunches of interest
AI <- rowSums( SBDI[,abu] )
sub <- t.in.range(tI, CS[, "t1"], CS[, "t2"]) & AI>0.1 & !is.na(AI)
Diff$Nm[i] <- mean( AI[sub] ) # average intensity
sub <- ti<=tI & tI<=tf
Diff$tN[[i]] <- tI[sub]
Diff$N[[i]] <- AI[sub]

# transverse sizes
```

```
# measurement times [h]
tabu <- c(paste("tSLASH",
                as.character(DS[i, "Bu1"]:DS[i, "Bu2"]), sep=""),
          paste("tSLASV",
                as.character(DS[i, "Bu1"]:DS[i, "Bu2"]), sep="") )
tte <- rowMeans( SL[,tabu] ) / 3600 + T0v["SL"]
# select bunches
asxSL <- paste("SLASH", as.character(DS[i, "Bu1"]:DS[i, "Bu2"]), sep="")
asySL <- paste("SLASV", as.character(DS[i, "Bu1"]:DS[i, "Bu2"]), sep="")
# average sizes [mm]
sxSL <- rowMeans( SL[, asxSL] )
sySL <- rowMeans( SL[, asySL] )
sub <- t.in.range(tte, CS[, "t1"], CS[, "t2"]) &
       sxSL>0 & sySL>0 & !is.na(sxSL) & !is.na(sySL)
Diff$sxSLm[i] <- mean( sxSL[sub] ) # beam rms [mm]
Diff$sySLm[i] <- mean( sySL[sub] )
sub <- ti<=tte & tte<=tf
Diff$tSL[[i]] <- tte[sub]
Diff$sxSL[[i]] <- sxSL[sub]
Diff$sySL[[i]] <- sySL[sub]

# momentum spread
# measurement times [h]
tabu <- c(paste("tSBDAMS",
                as.character(DS[i, "Bu1"]:DS[i, "Bu2"]), sep=""))
tme <- rowMeans( SBDM[, tabu] ) / 3600 + T0v["SBDM"]
# select bunches
ams <- paste("SBDAMS", as.character(DS[i, "Bu1"]:DS[i, "Bu2"]), sep="")
# average momentum spread [1E-4]
dpp <- rowMeans( SBDM[,ams] )
sub <- t.in.range(tme, CS[, "t1"], CS[, "t2"]) & dpp>0 & !is.na(dpp)
Diff$dppm[i] <- mean( dpp[sub] ) # momentum spread [1E-4]
sub <- ti<=tme & tme<=tf
Diff$tdpp[[i]] <- tme[sub]
Diff$dpp[[i]] <- dpp[sub]


# collimator location
cloc <- substring(DS[i, "CollName"], 1, 3)
```

```
# calculate beam sizes at collimator and at TEL2 [mm]
s1 <- c(rep(Diff$sxSLm[i], 2), rep(Diff$sySLm[i], 2))
deltap <- rep(Diff$dppm[i], 4)*1E-4
sigmas <-sigma2(sigma1 = s1,
         dpp = deltap,
         beta1 = LAT["aSL", c("betax", "betax", "betay", "betay")],
         beta2 = diag(as.matrix(LAT)[c(cloc, "TEL2", cloc, "TEL2"),
                     c("betax", "betax", "betay", "betay")]),
         D1 = LAT["aSL", c("Dx", "Dx", "Dy", "Dy")]*1E3,
         D2 = diag(as.matrix(LAT)[c(cloc, "TEL2", cloc, "TEL2"),
                  c("Dx", "Dx", "Dy", "Dy")])*1E3 )
Diff$sxC[i] <- sigmas[1]
Diff$sxTEL2[i] <- sigmas[2]
Diff$syC[i] <- sigmas[3]
Diff$syTEL2[i] <- sigmas[4]

Diff$ris[i] <- Diff$ri[i] / Diff$syTEL2[i]
Diff$ros[i] <- Diff$ro[i] / Diff$syTEL2[i]

# retrieve collimator positions
if(DS[i, "CollName"]=="F48V") {
  tC <- CP$tF48VCP/3600+T0v["CP"]
  cpmil <- CP$F48VCP}
if(DS[i, "CollName"]=="F49V") {
  tC <- CP$tF49VCP/3600+T0v["CP"]
  cpmil <- CP$F49VCP}
cpmm <- (cpmil-DS[i, "AxisPos"])*0.0254
sub <- ti<=tC & tC<=tf
Diff$tCP[[i]] <- tC[sub]
Diff$CP[[i]] <- cpmil[sub]

# retrieve local losses at collimator
if( DS[i, "LossMon"]=="S" ) { # total losses, slow logger
  tLoh <- CLb$tLF48 / 3600 + T0v["LO"]
  Lo <- CLb$LF48
  sub <- ti<=tLoh & tLoh<=tf
  Diff$tLo[[i]] <- tLoh[sub]
  Diff$Lo[[i]] <- Lo[sub]
}
if( DS[i, "LossMon"]=="T" ) { # total losses, fast logger
```

```
  tLoh <- TLF480_15Hz$tLF48 / 3600 + T0v["LO"]
  Lo <- TLF480_15Hz$LF48
  sub <- ti<=tLoh & tLoh<=tf
  Diff$tLo[[i]] <- tLoh[sub]
  Diff$Lo[[i]] <- Lo[sub]
}
if( DS[i, "LossMon"]=="G" ) { # gated loss monitors
  train <- floor((DS[i, "Bu1"]-1)/12) + 1 # bunch train of interest
  tLoh <- switch(train, # times
                 TLF49T_15Hz$tLF49T1,
                 TLF49T_15Hz$tLF49T2,
                 TLF49T_15Hz$tLF49T3) / 3600 + T0v["LO"]
  Lo <- switch(train, # losses
               TLF49T_15Hz$LF49T1,
               TLF49T_15Hz$LF49T2,
               TLF49T_15Hz$LF49T3)
  sub <- ti<=tLoh & tLoh<=tf
  Diff$tLo[[i]] <- tLoh[sub]
  Diff$Lo[[i]] <- Lo[sub]
}

}

# save global thermonuclear data container
setwd(default.dir)
save(Diff, file=output.fn)

return(Diff)
}

# collect fresh data or used saved data?
if(is.numeric(DATA.SOURCE))
  Diff <- CollectData(default.dir=ddir,
                      datasubset=DATA.SOURCE) else load(DATA.SOURCE)



# General
Table <- data.frame(
  ID = Diff$ID,
  Store = Diff$Sto,
```

```
  Date = paste(Diff$Da, month.abb[Diff$Mo], Diff$Yr),
  Time = paste(formatC(Diff$HH, width=2, flag="0"), ":",
        formatC(Diff$MM, width=2, flag="0"), sep=""),
  Collimator = Diff$CollName,
  BeamCenter = Diff$AxisPos,
  LossMon = Diff$LossMon,
  elensI = Diff$eIin,
  bunches = paste("A", Diff$Bu1, "--A", Diff$Bu2, sep=""),
  Notes = Diff$Notes
 )
T <- xtable(Table,
            caption="Data set summary.",
            label="tab:data_sets",
            align="llrllrcrrrl",
            digits=c(rep(NA, 2), 4, rep(NA, 3), 3, NA, 4, rep(NA, 2)),
            display=c(rep("s",2), "d", rep("s", 3), "fg", "s", "fg",
                      rep("s", 2)) )
print(T, floating.environment="table*",
      latex.environments=c("ruledtabular"), size="small", hline.after=0,
      include.rownames=FALSE)


# Electron lens
Table <- data.frame(
  ID = Diff$ID,
  Date = paste(Diff$Da, month.abb[Diff$Mo], Diff$Yr),
  Time = paste(formatC(Diff$HH, width=2, flag="0"), ":",
        formatC(Diff$MM, width=2, flag="0"), sep=""),
  Bg = Diff$Bg,
  Bm = Diff$Bm,
  eVm = Diff$eVm,
  eIm = Diff$eIm,
  eIin = Diff$eIin,
  rig = Diff$rig,
  rog = Diff$rog,
  ri = Diff$ri,
  ris = Diff$ris,
  ro = Diff$ro,
  ros = Diff$ros
 )
```

```
T <- xtable(Table,
            caption="Summary of electron lens conditions.",
            label="tab:elens_conditions",
            align="llllrrrrrrrrrrr",
            digits=c(rep(NA, 4), rep(3, 11)),
            display=c(rep("s", 4), rep("fg", 11)) )
print(T, floating.environment="table*",
      latex.environments=c("ruledtabular"), size="small", hline.after=0,
      include.rownames=FALSE)


# Beam conditions
Table <- data.frame(
  ID = Diff$ID,
  Date = paste(Diff$Da, month.abb[Diff$Mo], Diff$Yr),
  Time = paste(formatC(Diff$HH, width=2, flag="0"), ":",
        formatC(Diff$MM, width=2, flag="0"), sep=""),
  bunches = paste("A", Diff$Bu1, "--A", Diff$Bu2, sep=""),
  Nm = Diff$Nm,
  dppm = Diff$dppm,
  sxSLm = Diff$sxSLm,
  sxC = Diff$sxC,
  sxTEL2 = Diff$sxTEL2,
  sySLm = Diff$sySLm,
  syC = Diff$syC,
  syTEL2 = Diff$syTEL2
 )
T <- xtable(Table,
            caption="Summary of beam conditions.",
            label="tab:beam_conditions",
            align="llllrrrrrrrr",
            digits=c(rep(NA, 5), rep(3, 8)),
            display=c(rep("s", 5), rep("fg", 8)) )
print(T, floating.environment="table*",
      latex.environments=c("ruledtabular"), size="small", hline.after=0,
      include.rownames=FALSE)


# Lattice
Table <- Diff$LAT
```

```
colnames(Table) <- c("\\beta_x~\\mathrm{[m]}", "\\beta_y~\\mathrm{[m]}",
                     "D_x~\\mathrm{[m]}", "D_y~\\mathrm{[m]}")
rownames(Table)[4:5] <- c("$p$ SL", "$\\bar{p}$ SL")
T <- xtable(Table,
            caption="Tevatron lattice functions at the collimators (F48 and F49), the electron lens (TEL2), and (an
            label="tab:lattice_functions",
            align="ldddd",
            digits=c(NA, rep(4, 4)),
            display=c("s", rep("fg", 4)) )
print(T, floating.environment="table*",
      latex.environments=c("ruledtabular"), size="small",
      hline.after=0, sanitize.text.function=idem)




AnalyzeSteps <- function(set.list=0,
                         step.list=0,
                         DEBUG=FALSE){

nsets <- length(Diff$ID)

# global thermonuclear results container
Res <- list(
  # global parameters
  # loss rate calibration
  sigma.thr = numeric(nsets), # collimator is leading edge
  tt = list(), # resampled time
  dNm = list(), # average no. of lost particles vs. time
  ILm = list(), # average loss rate vs. time
  Kg = numeric(nsets), # calibration factor for losses [LM units / (particles/s)]
  eKg = numeric(nsets), # uncertainty
  Kg0 = numeric(nsets), # calibration offset
  eKg0 = numeric(nsets), # uncertainty
  Bg = numeric(nsets), # background level
  # see JINST paper on collimator system
#  dtstep = 0.277, # approx. time required to make a 50-um collimator step [s]
  dtstep = 0.040, # approx. time required to make a 50-um collimator step [s]
  # collimator parameters
  dtg = list(), # estimated time to make current collimator step [s]
  cpbef = list(), # coll. pos. before [mils]
```

```
cpaft = list(), # coll. pos. after [mils]
stepin = list(), # stepping inward? [logical]
xci = list(), # coll. pos. before [um]
xcf = list(), # coll. pos. after [um]
xsi = list(), # average coll. pos. [sigma]
Jci = list(), # coll. action before [um]
Jcf = list(), # coll. action after [um]
# intensity decay rate
lambda = numeric(nsets),
elambda = numeric(nsets),
# emittances and emittance growth rates
Wx = list(),
Wy = list(),
gammax = numeric(nsets),
egammax = numeric(nsets),
gammay = numeric(nsets),
egammay = numeric(nsets),
# core diffusion rates from emittance growth
D0x = numeric(nsets),
D0y = numeric(nsets),
eD0x = numeric(nsets),
eD0y = numeric(nsets),
# preliminary measurements and guesses
Nbef = list(), # number of circulating particles before step [1E9]
Lbef = list(), # loss level before
Laft = list(), # loss level after
Lrmsbef = list(), # fluctuations in losses before
Lrmsaft = list(), # fluctuations in losses after
Nlost = list(), # lost particles beam intensity
eNlost = list(), # uncertainty
ILlost = list(), # lost particles from loss-rate integral
Dg.pk = list(), # diffusion from peak [um^2/s]
Dg.tr = list(), # diffusion from transient [um^2/s]
eDg.tr = list(), # uncertainty on diffusion from transient [um^2/s]
Dg = list(), # final guess for diffusion coefficient before interp. [um^2/s]
lossrange = list(), # estimate of range of losses
DAig = list(), # guess of initial steady-state loss level [particles/s]
DAfg = list(), # guess of final steady-state loss level [particles/s]
# fit info and results
tfit = list(), # times [s]
```

```
  yfit = list(), # response [arb. units]
  wfit = list(), # weights
  rfit = list(), # residuals
  dof = list(), # degrees of freedom
  chi2 = list(), # chi-squared
  fit = list(), # model
  conv = list(), # convergence?
  hess = list(), # Hessian matrix
  errm = list(), # covariance matrix
  corm = list(), # correlation matrix
  pboot = list(), # bootstrap estimate of parameter distribution
  D = list(), # diffusion rate [um^2/s]
  eD = list(), # estimate of standard error
  DAi = list(), # initial steady-state loss level [particles/s]
  eDAi = list(), # estimate of standard error
  DAf = list(), # final steady-state loss level [particles/s]
  eDAf = list(), # estimate of standard error
  dtf = list(), # fitted time to move collimator [s]
  edtf = list(), # estimate of standard error
  t0f = list(), # fitted collimator movement time [s]
  et0f = list() # estimate of standard error
)

# general parameters
Kg <- 1
Bg <- 0
dtstep <- Res$dtstep

if(set.list==0) set.list <- seq(along=Diff$ID) # analyze them all

for(i in set.list) {

message("Analyzing data set ", Diff$ID[[i]], "\n")

# local losses at collimator
tLoh <- Diff$tLo[[i]]
Lo <- Diff$Lo[[i]]

# collimator parameters
xcmil <- Diff$AxisPos[i] # mils, approxiamte beam center axis
```

```
cloc <- substring(Diff$CollName[i], 1, 3) # collimator location
betac <- Diff$LAT[cloc, "betay"] # amplitude function [m]
sigmav <- Diff$syC[i] * 1E3 # beam sigma at collimator [um]

# collimator step information
tstart <- Diff$StepTimes[[i]][, "t1"]
tstep <- Diff$StepTimes[[i]][, "ts"]
tstop <- Diff$StepTimes[[i]][, "t2"]
nsteps <- length(tstep)

# collimator positions
tC <- Diff$tCP[[i]]
cpmil <- Diff$CP[[i]]
cpbef <- numeric(nsteps) # before and after each step
cpaft <- numeric(nsteps)
stepin <- logical(nsteps) # inward?
for(j in 1:nsteps){
  cpbef[j] <- median(
    cpmil[ (tstep[j]-60/3600) <= tC & tC <= (tstep[j]-15/3600) ], na.rm=TRUE)
  cpaft[j] <- median(
    cpmil[ (tstep[j]+15/3600) <= tC & tC <= (tstep[j]+60/3600) ],
                     na.rm=TRUE )
  stepin[j] <- FALSE
  if(Diff$CollName[[i]]=="F48V"){
    if(cpaft[j] > cpbef[j]) stepin[j] <- TRUE} # in=>larger value for F48V
  if(Diff$CollName[[i]]=="F49V"){
    if(cpaft[j] < cpbef[j]) stepin[j] <- TRUE} # in=>smaller value for F49V
}
if(DEBUG) message("Step-in =\n", as.numeric(stepin))
dtg <- dtstep*abs(cpbef-cpaft)/2 # approximate time [s] to make the step

Res$cpbef[[i]] <- cpbef
Res$cpaft[[i]] <- cpaft
Res$stepin[[i]] <- stepin
Res$dtg[[i]] <- dtg

# collimator positions and actions w.r.t. the beam center
Res$xci[[i]] <- xci <- (cpbef-xcmil)*um.mil # [um]
Res$xcf[[i]] <- xcf <- (cpaft-xcmil)*um.mil
xc <- 0.5*(xci+xcf)
```

```
Res$xsi[[i]] <- xsi <- xc/sigmav # in units of sigma
dxc <- abs(xci-xcf)
Res$Jci[[i]] <- Jci <- xci*xci/betac / 1e6 # action at collimator [um]
Res$Jcf[[i]] <- Jcf <- xcf*xcf/betac / 1e6
Jcc <- xc*xc/betac / 1e6
deltaJ <- abs(Jci-Jcf)

# guess parameters
Nbef <- numeric(nsteps) # total number of circulating particles before step [1E9]
Lbef <- numeric(nsteps) # loss level before
Laft <- numeric(nsteps) # loss level after
Lrmsbef <- numeric(nsteps) # fluctuations in losses before
Lrmsaft <- numeric(nsteps) # fluctuations in losses after
Nlost <- numeric(nsteps)
eNlost <- numeric(nsteps)
ILlost <- numeric(nsteps)
Dg.pk <- numeric(nsteps) # diffusion from peak [um^2/s]
Dg.tr <- numeric(nsteps) # diffusion from transient [um^2/s]
eDg.tr <- numeric(nsteps) # uncertainty on diffusion from transient [um^2/s]
Dg <- numeric(nsteps) # input parameter for interpolation
lossrange <- numeric(nsteps) # estimate of range of losses
DAig <- numeric(nsteps)
DAfg <- numeric(nsteps)
# fit parameters
D <- numeric(nsteps)
eD <- numeric(nsteps)
DAi <- numeric(nsteps)
eDAi <- numeric(nsteps)
DAf <- numeric(nsteps)
eDAf <- numeric(nsteps)
dtf <- numeric(nsteps)
edtf <- numeric(nsteps)
t0f <- numeric(nsteps)
et0f <- numeric(nsteps)
# goodness of fit
tfit <- list(nsteps)
yfit <- list(nsteps)
wfit <- list(nsteps)
rfit <- list(nsteps)
current.fits <- list(nsteps)
```

```
conv <- logical(nsteps)
fitDOF <- numeric(nsteps)
fitChi2 <- numeric(nsteps)
hess <- list(nsteps)
errm <- list(nsteps)
corm <- list(nsteps)
pboot <- list(nsteps)

# acceptable diffusion rates
Ddefault <- 1E-6
Dmin <- 1E-14
Dmax <- 1E4
lDrange <- 4 # +/- search range for log10(D)

# smooth-spline fit of number of particles
# for step-by-step loss-rate calibration
if(DEBUG) message("Interpolating intensity...")
tN <- Diff$tN[[i]]
N <- Diff$N[[i]]
Nsm <- smooth.spline(tN, N)

# decay rate [1/s], collimators farther than sigma.thr sigmas
if(DEBUG) message("Calculating intensity decay rate...")
sigma.thr <- min(max(abs(Res$xsi[[i]]))+1, 6)
tc <- Diff$tCP[[i]]
yc <- abs(Diff$CP[[i]]-xcmil)*um.mil/sigmav
TI <- FindInterval(t=tc, x=yc, co=yc>sigma.thr)
tau1 <- TI$ti; tau2 <- TI$tf
sub <- tau1<Diff$tN[[i]] & Diff$tN[[i]]<tau2
tN <- Diff$tN[[i]][sub]
tNs <- (tN-tN[1])*3600
N <- Diff$N[[i]][sub]
mlambda <- nls(N ~ N0*exp(-lambda.par*tNs),
               start=list(N0=median(N), lambda.par=IQR(N)/diff(range(tNs))),
               control=list(warnOnly=TRUE))
Rl <- summary(mlambda)$coefficients
lambda <- Rl["lambda.par", 1]
elambda <- Rl["lambda.par", 2]
Res$lambda[i] <- lambda
Res$elambda[i] <- elambda
```

```
if(DEBUG) {
  message("\tlambda = ", formatC(lambda, di=4), " +/- ",
          formatC(elambda, di=3), " 1/s\n\ttau = ",
          formatC(1/lambda/3600, di=4), " h") }

# calibration of loss rates
# integral of losses = k * (# of lost particles)
if(DEBUG) message("Calibrating losses...")
# find time interval for which collimator defines aperture
sigma.thr <- max(min(abs(Res$xsi[[i]]))+1, 6) # chosen threshold
if(DEBUG) message("Chosen sigma threshold = ", sigma.thr)
tc <- Diff$tCP[[i]]
yc <- abs(Diff$CP[[i]]-xcmil)*um.mil/sigmav
TI <- FindInterval(t=tc, x=yc, co=yc<sigma.thr)
tau1 <- TI$ti; tau2 <- TI$tf
# exclude tests of gated loss monitor delays
if(Diff$ID[[i]] %in% c("8749.hi.T1", "8749.hi.tel", "8749.hi.T3")){
  if(DEBUG) message("Excluding gated loss monitor tests...")
  ttest1 <- 7.971; ttest2 <- 8.096
  if(tau1<ttest1 & tau2>ttest2){ # straggling both, choose largest interval
    if(abs(tau1-ttest1) > abs(tau2-ttest2)) tau2<-ttest1 else tau1<-ttest2}
  if(tau1<ttest1 & tau2>ttest1) tau2 <- ttest1
  if(tau1<ttest2 & tau2>ttest2) tau1 <- ttest2
}
if(DEBUG) message("Chosen time interval: ", tau1, "\t", tau2)
sub <- tau1<Diff$tN[[i]] & Diff$tN[[i]]<tau2
tN <- Diff$tN[[i]][sub]
N <- Diff$N[[i]][sub]
# integrated losses
sub <- tau1<Diff$tLo[[i]] & Diff$tLo[[i]]<tau2
tL <- Diff$tLo[[i]][sub]
L <- Diff$Lo[[i]][sub]
IL <- cumsum(L) * mean(diff(Diff$tLo[[i]][sub])*3600, na.rm=TRUE)
# resample times
nt <- min(length(tN), length(tL)) / 16
tNf <- cut(tN, breaks=seq(tau1, tau2, length=nt))
tLf <- cut(tL, breaks=seq(tau1, tau2, length=nt))
tt <- tapply(tL, tLf, mean, na.rm=TRUE)
# average losses
Nm <- tapply(N, tNf, mean, na.rm=TRUE)
```

```
ILm <- tapply(IL, tLf, mean, na.rm=TRUE)
dNm <- (mean(Nm[1:4])-Nm)*1e9 # lost particles
# calibration factors
mcal <- lm(ILm ~ dNm)
Kg <- coef(mcal)["dNm"]
eKg <- summary(mcal)$coef[2,2]
Kg0 <- summary(mcal)$coef[1,1]
eKg0 <- summary(mcal)$coef[1,2]
if(DEBUG){
  dev.new()#; layout(matrix(1:2, nrow=2))
  plot(dNm, ILm)
  abline(mcal, col="red")
#  plot(tt, dNm/ILm)
  message("Calibration factor Kg = ", Kg)
  message("Intercept Kg0 = ", Kg0)
}
if(Kg<=0){Kg <- 1; eKg <- 0.1}
#if(1){Kg <- 1; eKg <- 0.1}

# synchrotron-light emittances and emittance growth rates
if(DEBUG) message("Calculating emittance growth rates...")
betax <- Diff$LAT["aSL", "betax"] # [m]
betay <- Diff$LAT["aSL", "betay"] # [m]
Dx <- Diff$LAT["aSL", "Dx"] * 1e3 # [mm]
Dy <- Diff$LAT["aSL", "Dy"] * 1e3 # [mm]
tW <- Diff$tSL[[i]] # [h]
tWs <- (tW-tW[1])*3600
dpp.sm <- smooth.spline(Diff$tdpp[[i]], Diff$dpp[[i]])
dpp <- predict(dpp.sm, x=tW)$y * 1e-4 # fraction
Wx <-( Diff$sxSL[[i]]^2 - (Dx*dpp)^2 ) / betax
Wy <-( Diff$sySL[[i]]^2 - (Dy*dpp)^2 ) / betay
mWx <- nls(Wx ~ W0 * exp(gamma*tWs),
           start=list(W0=median(Wx), gamma=IQR(Wx)/diff(range(tWs))),
           control=list(warnOnly=TRUE))
mWy <- nls(Wy ~ W0 * exp(gamma*tWs),
           start=list(W0=median(Wy), gamma=IQR(Wy)/diff(range(tWs))),
           control=list(warnOnly=TRUE))
Rx <- summary(mWx)$coefficients
Ry <- summary(mWy)$coefficients
gammax <- Rx["gamma", 1]
```

```
egammax <- Rx["gamma", 2]
gammay <- Ry["gamma", 1]
egammay <- Ry["gamma", 2]
Res$Wx[[i]] <- Wx
Res$Wy[[i]] <- Wy
Res$gammax[i] <- gammax
Res$egammax[i] <- egammax
Res$gammay[i] <- gammay
Res$egammay[i] <- egammay
if(DEBUG) message("\th: ", formatC(gammax, di=4), " +/- ",
  formatC(egammax, di=3), " 1/s\n\tv: ", formatC(gammay, di=4), " +/- ",
  formatC(egammay, di=3), " 1/s")

# core diffusion rates from emittance growth
if(DEBUG) message("Calculating core diffusion rates...")
D0x <- 2 * median(Wx)^2 * gammax
D0y <- 2 * median(Wy)^2 * gammay
eD0x <- 2 * median(Wx)^2 * egammax
eD0y <- 2 * median(Wy)^2 * egammay
Res$D0x[i] <- D0x
Res$D0y[i] <- D0y
Res$eD0x[i] <- eD0x
Res$eD0y[i] <- eD0y
if(DEBUG) message("\th: ", formatC(D0x, di=4), " +/- ",
  formatC(eD0x, di=3), " um^2/s\n\tv: ", formatC(D0y, di=4), " +/- ",
  formatC(eD0y, di=3), " um^2/s")

# Analyze steps
# choose subset of steps to analyze
if(step.list[1]==0)
  explist <- seq(along=tstart) else explist <- step.list

for(ex in explist){
  message("Analyzing step #", ex)
  # number of particles
  befN <- tstart[ex] <= Diff$tN[[i]] & Diff$tN[[i]] <= (tstep[ex]-0.2/3600)
  Nbef[ex] <- mean(Diff$N[[i]][befN], na.rm=TRUE)
  # lost particles from smooth spline interpolation
  subN <- tstart[ex] <= Diff$tN[[i]] & Diff$tN[[i]] <= tstop[ex]
  Nlost[ex] <- diff(predict(Nsm, x=c(tstop[ex], tstart[ex]))$y)*1e9
```

```
  # uncertainty on number of lost particles from smooth spline
  re <- residuals(Nsm)[subN]
  eNlost[ex] <- sqrt(2) * mad(re) / sqrt(length(re)) *1e9
  if(DEBUG){
    xx <- Diff$tN[[i]][subN]
    yy <- Diff$N[[i]][subN]
    dev.new(); plot(xx, yy); points(Nsm, type="l", col="red")
    message("Lost particles: ", formatC(Nlost[ex], di=4),
            " +/- ", formatC(eNlost[ex], di=3)) }
  # define regions: before, transient, after
  bef <- tstart[ex] <= tLoh & tLoh <= (tstep[ex]-0.2/3600)
  aft <- max(tstep[ex], tstop[ex]-20/3600) <= tLoh & tLoh <= tstop[ex]
  all <- tstart[ex] <= tLoh & tLoh <= tstop[ex]
  # data of interest for model
  sub <- tstart[ex] <= tLoh & tLoh <= tstop[ex]
#  sub <- (tstep[ex]+1/3600) <= tLoh & tLoh <= tstop[ex]
  x <- (tLoh[sub]-tstep[ex])*3600 # time since step [s]
  y <- Lo[sub]
  if(DEBUG) {dev.new(); plot(x, y)}
  # loss levels and fluctuations
  Lbef[ex] <- mean(Lo[bef], na.rm=TRUE)
  Laft[ex] <- mean(Lo[aft], na.rm=TRUE)
  if(abs(tstop[ex]-tstep[ex]) < 20/3600) Laft[ex] <- Lbef[ex]
  Lrmsbef[ex] <- mad(Lo[bef], na.rm=TRUE)
  Lrmsaft[ex] <- mad(Lo[aft], na.rm=TRUE)
  if(Lrmsbef[ex] <= 0 | !is.finite(Lrmsbef[ex]))
     Lrmsbef[ex] <- sd(Lo[bef], na.rm=TRUE)
  if(Lrmsbef[ex] <= 0 | !is.finite(Lrmsbef[ex])) Lrmsbef[ex] <- Lbef[ex]
  if(Lrmsaft[ex] <= 0 | !is.finite(Lrmsaft[ex]))
     Lrmsaft[ex] <- sd(Lo[aft], na.rm=TRUE)
  if(Lrmsaft[ex] <= 0 | !is.finite(Lrmsaft[ex])) Lrmsaft[ex] <- Laft[ex]
  if(DEBUG) message("Lbef = ", Lbef[ex], ", rms ", Lrmsbef[ex])
  if(DEBUG) message("Laft = ", Laft[ex], ", rms ", Lrmsaft[ex])
  # integrated losses
  sub <- tstep[ex]<=tLoh & tLoh<=tstop[ex] # region of interest
  deltat <- mean(diff(tLoh[sub]))*3600
  ILlost[ex] <- sum(Lo[sub], na.rm=TRUE)*deltat
  # exclude or correct funky behavior of gated losses (saturation?)
  FUNKY <- FALSE
  dtexcl <- 3 # [s]
```

```
  if(Diff$LossMon[[i]]=="G" &
     Res$stepin[[i]][ex]==TRUE &
     as.numeric(quantile(y, probs=0.99)) > 3e5) {
       sub1 <- 0<x & x<=1
       m1 <- mean(y[sub1], na.rm=TRUE)
       if( m1<Laft[ex] ) FUNKY <- TRUE
  }
  # estimate peak or dip
  gt0 <- tstep[ex] < tLoh & tLoh <= tstop[ex]
#  lora <- quantile(Lo[gt0], probs=c(.01,.99), na.rm=TRUE)
  lora <- range(Lo[gt0], na.rm=TRUE)
  lossrange[ex] <- max(abs(lora-Lbef[ex]))
  lossrange[ex] <- abs(lossrange[ex] - abs(Lbef[ex]-Laft[ex]))
  if(DEBUG) message("Loss range = ", lossrange[ex])
  # diffusion rate estimate from peak/dip value
  Dg.pk[ex] <- Ddefault
  if( (dtg[ex]>0) & (Lbef[ex]>0) & (lossrange[ex]>0) )
    Dg.pk[ex] <- ( deltaJ[ex]*Lbef[ex]/lossrange[ex] )^2 / pi / dtg[ex]
  if(DEBUG) message("Dg.pk = ", formatC(Dg.pk[ex], di=4))
  # diffusion rate estimate from simplfied model applied to transient
  Dg.tr[ex] <- Ddefault
  eDg.tr[ex] <- 0.1*Ddefault
  # define transient region
  if(FUNKY){
    tra <- (tstep[ex]+dtexcl/3600)<=tLoh & tLoh<=tstop[ex]
  }else{
    tra <- (tstep[ex]+(dtg[ex]+0.5)/3600)<=tLoh & tLoh<=tstop[ex] }
  xx <- (tLoh[tra]-tstep[ex])*3600 # times in s
  yy <- Lo[tra]
  if(stepin[ex]) sgn<-+1 else sgn<--1
  ww <- Laft[ex] + Laft[ex]*sgn*deltaJ[ex]/sqrt(pi*xx)
  if(DEBUG) {dev.new(); plot(ww, yy)}
  mtr <- lm(yy ~ ww)
  inv.sqrt.DD <- coef(mtr)["ww"]
  ei <- summary(mtr)$coef[1,2]
#  ei <- diff(confint(mtr, parm="ww", level=std.lvl))/2
  if(DEBUG) message("Linear slope: ", inv.sqrt.DD, " +/-", ei)
  if(DEBUG) abline(mtr, col="red")
  if(inv.sqrt.DD!=0){
    Dg.tr[ex] <- 1/inv.sqrt.DD^2
```

```
    eDg.tr[ex] <- abs(0.5*ei/inv.sqrt.DD^1.5)}
  if(DEBUG) message("Dg.tr = ", formatC(Dg.tr[ex], di=4),
                    " +/- ", formatC(eDg.tr[ex], di=3))
  # diffusion rate guess
  D[ex] <- Dg[ex] <- Ddefault
  eD[ex] <- 0.1*Ddefault
  if( Dmin<=Dg.pk[ex] & Dg.pk[ex]<=Dmax &
      Dmin<=Dg.tr[ex] & Dg.tr[ex]<=Dmax   ){
    rD <- range(log10(c(Dg.pk[ex], Dg.tr[ex])))
#    D[ex] <- Dg[ex] <- 10^((.8)*rD[1] + (.2)*rD[2]) # more weight to small one
#    D[ex] <- Dg[ex] <- sqrt(Dg.pk[ex]*Dg.tr[ex]) # geometrical mean
    # trust transient more
    D[ex] <- Dg[ex] <-
        10^((0.2)*log10(Dg.pk[ex]) + (0.8)*log10(Dg.tr[ex]))
#    D[ex] <- Dg[ex] <- Dg.pk[ex]
#    D[ex] <- Dg[ex] <- Dg.tr[ex]
#    D[ex] <- Dg[ex] <- 1E-7
#    D[ex] <- Dg[ex] <- mean(Dg.pk[ex], Dg.tr[ex])
#    D[ex] <- Dg[ex] <- min(Dg.pk[ex], Dg.tr[ex])
    eD[ex] <- abs(Dg.pk[ex]-Dg.tr[ex])/sqrt(12)}
  # guesses of slopes of distribution function
  DAi[ex] <- DAig[ex] <- abs(Lbef[ex])
  DAf[ex] <- DAfg[ex] <- abs(Laft[ex])
  if(Lbef[ex]>0)
    eDAi[ex] <- Lrmsbef[ex]*DAig[ex]/Lbef[ex] else eDAi[ex] <- 0.1*DAig[ex]
  if(Laft[ex]>0)
    eDAf[ex] <- Lrmsaft[ex]*DAfg[ex]/Laft[ex] else eDAf[ex] <- 0.1*DAfg[ex]
  # guesses for collimator step time
  t0f[ex] <- 0
  et0f[ex] <- 1/20/sqrt(12)
  dtf[ex] <- dtg[ex]
  edtf[ex] <- 1/20/sqrt(12)
  # define weights
  s <- mad(y, na.rm=TRUE)
  if(s<=0 | !is.finite(s)) s <- sd(y, na.rm=TRUE)
  if(s<=0 | !is.finite(s)) s <- 1
  wei <- rep(1/s, length(y))
#  loc.rms <- LocalRMS(x=y, n=40)  # local rms of losses
#  wei <- 1/(loc.rms)^2
  rwei <- range(wei)
```

```
  if(DEBUG) message("Range of weights is [", formatC(rwei[1], di=3),
  ",", formatC(rwei[2], di=3), "]" )
  # exclude or correct funky behavior of gated losses (saturation?)
  if(FUNKY){
    sub1 <- 0<=x & x<=dtexcl
#    wei[sub1] <- 0 # exclude
    wei[sub1] <- wei[sub1]/2 # reduce weights
    y[sub1] <- Laft[ex] + abs(y[sub1]-Laft[ex]) # correct
    if(DEBUG) {dev.new(); plot(x, y)}
  }
  # minimize sum of deviations
  # need to check distribution of response (Poisson+oscillations?)
  fom <- function(p){ # figure-of-merit to minimize
    ye <- loss.rate(t=x, K=1, B=Bg,
                    Jci=Jci[ex], Jcf=Jcf[ex],
                    D=10^(p[1]),
                    Ai=10^(p[2]-p[1]), Af=10^(p[3]-p[1]),
                    t0=t0f[ex],
                    dt=dtf[ex])
    fom <- mean( abs(y-ye) , na.rm=TRUE )
#    fom <- mean( log(1+0.5*(wei*(y-ye))^2) , na.rm=TRUE )
    return(fom/s)}
  chisqr <- function(p){ # chi-square for error estimation
    ye <- loss.rate(t=x, K=1, B=Bg,
                    Jci=Jci[ex], Jcf=Jcf[ex],
                    D=10^(p[1]),
                    Ai=10^(p[2]-p[1]), Af=10^(p[3]-p[1]),
                    t0=t0f[ex],
                    dt=dtf[ex])
    chisqr <- sum( wei*wei*(y-ye)*(y-ye) , na.rm=TRUE )
    return(chisqr)}
  # parameter definitions
  if(DEBUG) message("Initial guesses:\n\tD = ", Dg[ex],
                    "\n\tDAi = ", DAig[ex],
                    "\n\tDAf = ", DAfg[ex],
                    "\n\tt0 = ", t0f[ex],
                    "\n\tdt = ", dtf[ex])
  pnam <- c("lD", "lDAi", "lDAf") # logarithms
  ipar <- log10(c(Dg[ex], DAig[ex], DAfg[ex])) # initial values
  npar <- length(ipar)
```

```
  lop <- ipar + c(-lDrange, rep(-0.5, 2)) # lower limits
  upp <- ipar + c(+lDrange, rep(+0.5, 2)) # upper limits
  pars <- data.frame(initial=ipar, lower=lop, upper=upp) # parameter summary
  rownames(pars) <- pnam
  # save model inputs
  tfit[[ex]] <- x
  yfit[[ex]] <- y
  wfit[[ex]] <- wei
  # FUNCTION MINIMIZATION
  fomv <- function(x) fom(p=c(x, ipar[2:3]))
  fomv <- Vectorize(fomv, "x")
  if(DEBUG){ dev.new(); curve(fomv(x), ipar[1]-lDrange, ipar[1]+lDrange, log="y") }
  if(DEBUG) message("Fitting with ", npar, " parameter(s)")
  # preliminary minimization with one variable
  lD.test <- seq(lop[1], upp[1], by=0.05)
  fom.test <- fomv(lD.test)
  i.test <- which.min(fom.test)
  Dg[ex] <- 10^(lD.test[i.test])
  if(DEBUG) message("Dg = ", formatC(Dg[ex], di=4))
#  fit <- optim(par=ipar[1], fn=fomv, method="Brent",
#               lower=lop[1], upper=upp[1])
#  Dg[ex] <- 10^(fit$par)
  # choose best value as starting point
  lD.test <- log10( c(Dg.pk[ex], Dg.tr[ex], Dg[ex]) )
  fom.test <- fomv(lD.test)
  i.test <- which.min(fom.test)
  ipar[1] <- lD.test[i.test]
  pars$lower[1] <- lop[1] <- ipar[1]-lDrange
  pars$upper[1] <- upp[1] <- ipar[1]+lDrange
  D[ex] <- 10^(ipar[1])
  # final minimization
  fit <- optim(par=ipar, fn=fom, hessian=TRUE
#               , method="BFGS"
               , method="L-BFGS-B", lower=lop, upper=upp
               )
  current.fits[[ex]] <- fit
  pars$fitted <- pa <- fit$par
  pars$errors <- er <- rep(NA, npar)
  chi2.properties <- fdHess(pa, chisqr)
  fitChi2[ex] <- chi2 <- chisqr(pa)
```

```
#  fitChi2[ex] <- chi2 <- fit$value
#  fitChi2[ex] <- chi2 <- chi2.properties$mean
  fitDOF[ex] <- dof <- length(x)-npar
#  hess[[ex]] <- H <- fit$hessian
  hess[[ex]] <- H <- chi2.properties$Hessian
  errm[[ex]] <- emat <- 0*H
  corm[[ex]] <- cmat <- 0*H
  if(det(H)!=0) {
#    errm[[ex]] <- emat <- solve(H, tol=1e-24) # covariance matrix
    errm[[ex]] <- emat <- chol2inv(chol(H, pivot=TRUE)) # covariance matrix
    if(any(diag(emat) < 0)) warning("Covariance matrix has negative elements on diagonal.")
    pars$errors <- er <- sqrt(abs(diag(emat))) # parameter errors
    corm[[ex]] <- cmat <- cov2cor(emat)
  }
  # number of synthetic data sets (18 -> 17% error on sigma, 50 -> 10%)
  nboot <- 18
  bootm <- matrix(nrow=npar, ncol=nboot) # save parameters for each experiment
  bootm[, 1] <- pa
  if(BOOTSTRAP){
    if(DEBUG) message("Bootstrapping...")
    npt <- length(x)
    for(jj in 2:nboot){
      kk <- sample(1:npt, npt, replace=TRUE)
      x <- tfit[[ex]][kk]
      y <- yfit[[ex]][kk]
      fit <- optim(par=ipar, fn=fom, method="L-BFGS-B", lower=lop, upper=upp)
      bootm[, jj] <- fit$par }
    pars$fitted <- pa <- apply(bootm, 1, median, na.rm=TRUE)
    pars$errors <- er <- apply(bootm, 1, mad, na.rm=TRUE)
    x <- tfit[[ex]]
    y <- yfit[[ex]] }
  pboot[[ex]] <- bootm
  if(DEBUG){
    message("\nParameters:\n")
    write.table(round(pars, 6), file=stderr(), sep="\t")
    message("\nHessian matrix:\n")
    write.matrix(H, file=stderr())
    message("\nCovariance matrix:\n")
    write.matrix(emat, file=stderr())
    message("\nCorrelation matrix:\n")
```

```
    write.matrix(round(cmat, 4), file=stderr())
  }
  if(DEBUG) message("Chi2 / dof = ", chi2, " / ", dof, " = ", chi2/dof)
  conv[ex] <- FALSE
  if(fit$convergence==0) conv[ex] <- TRUE
  if(DEBUG) message("Convergence code = ", fit$convergence)
  # save parameters
  lD <- pa[1]
  elD <- er[1]
  D[ex] <- 10^(lD)
  eD[ex] <- 0.5*abs(10^(lD+elD)-10^(lD-elD))
  lDA <- pa[2]
  elDA <- er[2]
  DAi[ex] <- 10^(lDA)
  eDAi[ex] <- 0.5*abs(10^(lDA+elDA)-10^(lDA-elDA))
  lDA <- pa[3]
  elDA <- er[3]
  DAf[ex] <- 10^(lDA)
  eDAf[ex] <- 0.5*abs(10^(lDA+elDA)-10^(lDA-elDA))
#  t0f[ex] <- pa[4]
#  et0f[ex] <- er[4]
#  dtf[ex] <- pa[5]
#  edtf[ex] <- er[5]
  FittedLossRate <- function(x) {
    LR <- 0*x
    LR <- loss.rate(t=x, K=1, B=Bg, t0=t0f[ex],
                    Ai=DAi[ex]/D[ex], Af=DAf[ex]/D[ex],
                    Jci=Jci[ex], Jcf=Jcf[ex],
                    D=D[ex], dt=dtf[ex])
    return(LR)}
  rfit[[ex]] <- y - FittedLossRate(x) # residuals
  if(DEBUG){
    cfit <<- fit
    xxm <- min(x, na.rm=TRUE); xxM <- max(x, na.rm=TRUE)
    if(xxm<0 & xxM>0){
      xx1 <- seq(xxm, xxm/10, length=5)
      xx2 <- seq(xxm/10, xxM/10, length=101)
      xx3 <- seq(xxM/10, xxM, length=41)
      xx <- c(xx1, xx2, xx3)} else {
      xx <- seq(xxm, xxM, length=41) }
```

```
#    yy <- predict(fit, newdata=list(x=xx))
    yy <- FittedLossRate(xx)
    dev.new(); plot(x, y)
    points(xx, yy, type="l", col="red")
    message("\nFit results:")
    message("\tD = ", D[ex], " +/- ", eD[ex])
    message("\tDAi = ", DAi[ex], " +/- ", eDAi[ex])
    message("\tDAf = ", DAf[ex], " +/- ", eDAf[ex])
#    message("\tt0 = ", t0f[ex], " +/- ", et0f[ex])
#    message("\tdt = ", dtf[ex], " +/- ", edtf[ex])
  }

} # end of loop over experiments

# copy results to global data structure
Res$sigma.thr[[i]] <- sigma.thr
Res$tt[[i]] <- tt
Res$dNm[[i]] <- dNm
Res$ILm[[i]] <- ILm
Res$Kg[i] <- Kg
Res$eKg[i] <- eKg
Res$Kg0[i] <- Kg0
Res$eKg0[i] <- eKg0
Res$Bg[i] <- Bg
Res$Nbef[[i]] <- Nbef
Res$Lbef[[i]] <- Lbef
Res$Laft[[i]] <- Laft
Res$Lrmsbef[[i]] <- Lrmsbef
Res$Lrmsaft[[i]] <- Lrmsaft
Res$Nlost[[i]] <- Nlost
Res$eNlost[[i]] <- eNlost
Res$ILlost[[i]] <- ILlost
Res$Dg.pk[[i]] <- Dg.pk
Res$Dg.tr[[i]] <- Dg.tr
Res$eDg.tr[[i]] <- eDg.tr
Res$Dg[[i]] <- Dg
Res$lossrange[[i]] <- lossrange
Res$DAig[[i]] <- DAig
Res$DAfg[[i]] <- DAfg
```

```
Res$tfit[[i]] <- tfit
Res$yfit[[i]] <- yfit
Res$wfit[[i]] <- wfit
Res$rfit[[i]] <- rfit

Res$fit[[i]] <- current.fits
Res$conv[[i]] <- conv
Res$dof[[i]] <- fitDOF
Res$chi2[[i]] <- fitChi2

Res$hess[[i]] <- hess
Res$errm[[i]] <- errm
Res$corm[[i]] <- corm
Res$pboot[[i]] <- pboot

Res$D[[i]] <- D
Res$eD[[i]] <- eD
Res$DAi[[i]] <- DAi
Res$eDAi[[i]] <- eDAi
Res$DAf[[i]] <- DAf
Res$eDAf[[i]] <- eDAf
Res$t0f[[i]] <- t0f
Res$et0f[[i]] <- et0f
Res$dtf[[i]] <- dtf
Res$edtf[[i]] <- edtf

} # end of loop over data sets

# save results
save(Res, file="tmp_Results.RData")
return(Res)
}

if(is.numeric(RESULTS.SOURCE))
  Res <- AnalyzeSteps(set.list=RESULTS.SOURCE,
                      step.list=STEPS.SUBSET,
                      DEBUG=DEBUG.MODE) else load(RESULTS.SOURCE)


message("Plotting calibration of losses...")
```

```
basename <- paste(Diff$ddir, "/", "tmp_loss_cal", sep="")
pdf(file=paste(basename, ".pdf", sep=""), width=6, height=8)
dev.control(displaylist="enable") # enable recording
par(oma=rep(0,4), mar=c(5.2,3.2,0.2,0.2), mgp=c(2,0.5,0), tcl=-0.25)

ID <- unlist(Diff$ID)
K <- unlist(Res$Kg)
eK <- unlist(Res$eKg)
NB <- Diff$Bu2-Diff$Bu1+1

j <- which(Diff$LossMon=="T")
idT <- ID[j]
kT <- K[j]
ekT <- eK[j]
nbT <- NB[j]
xT <- 1:length(kT)

j <- which(Diff$LossMon=="G")
idG <- ID[j]
kG <- K[j]
ekG <- eK[j]
nbG <- NB[j]
xG <- 1:length(kG)

layout(matrix(1:2, nrow=2))

PlotLossCal <- function(x=xT, y=kT, ey=ekT, nb=nbT, lab.id=idT,
                        mess="T:LF480") {
  nbcol <- "blue"
  plot(x, y, axes=FALSE, xlab="",
    ylab=expression(paste(K[g], " [(loss mon. units) / (particles/s)]", sep="")))
  arrows(x, y-ey, x, y+ey, le=0.03, angle=90, code=3)
  text(x, y, labels=as.character(nb), pos=4, col=nbcol)
  axis(1, at=x, labels=lab.id, las=3)
  axis(2)
  legend("topleft", bty="n", legend=mess)
  legend("bottomleft", bty="n", text.col=nbcol, legend="Number of bunches")
  box()
}
```

```
PlotLossCal()
PlotLossCal(x=xG, y=kG, ey=ekG, nb=nbG, lab.id=idG, mess="T:LF49Tn")

dev.copy2eps(file=paste(basename, ".eps", sep=""))
dev.off()
IncludePlotInLatex(file=basename, flo=FALSE,
                   cap="Calibration of losses.",
                   lab="fig:losscal")


message("Plotting total vs gated measurements...")
CompareSets(set.list=c(23, 24, 26), co=matlab.colors(3), jwhich=NA,
            plot.id="tot_vs_gated")


message("Plotting inward vs outward measurements (collisions)...")
CompareSets(set.list=c(8, 10, 16), co=matlab.colors(3), jwhich=NA,
            xlim.par=c(3.9, 14), plot.id="in_vs_out_coll")


message("Plotting inward vs outward measurements (p-bar only)...")
CompareSets(set.list=c(28), co="black", jwhich=NA,
            xlim.par=c(3.9, 14), plot.id="in_vs_out_pbar")


message("Plotting wide range (no TEL)...")
CompareSets(set.list=8, co="black", jwhich=NA, core=TRUE, plot.id="wide_range")


message("Plotting pbar-only...")
CompareSets(set.list=c(28, 31), co=matlab.colors(2), jwhich=NA,
            core=TRUE, plot.id="pbar_only")


message("Plotting collisions vs. pbar-only...")
CompareSets(set.list=c(3, 5, 8, 16, 28, 31), co=matlab.colors(6), jwhich=NA,
            core=TRUE, plot.id="coll_vs_pbar_wide")


message("Plotting collisions vs. pbar-only...")
```

```
CompareSets(set.list=c(3, 5, 8, 16, 28, 31), co=matlab.colors(6), jwhich=NA,
            core=TRUE, xlim.par=c(0, 14), plot.id="coll_vs_pbar_narrow")


message("Plotting diffusion vs e-lens position...")
CompareSets(set.list=c(8, 9, 13, 20, 21), co=matlab.colors(5), jwhich=NA,
            xlim.par=c(3.9, 10), elens=TRUE, plot.id="elens_pos")


message("Plotting diffusion vs e-lens intensity...")
CompareSets(set.list=c(16, 24, 25, 21, 17), co=matlab.colors(5), jwhich=NA,
            elens=TRUE, plot.id="elens_int")


message("Plotting diffusion with e-lens during pbar only...")
CompareSets(set.list=c(28, 31, 29, 32), co=matlab.colors(4), jwhich=NA,
            elens=TRUE, xlim.par=c(3.9, 12), plot.id="elens_pbar_only")



MakeDiagnosticPlots <- function(set.list=0){
wi<-6; he<-6; pt<-10; fo<-"Times"
ps.options(width=wi, height=he, pointsize=pt, family=fo, bg="white")
pdf.options(width=wi, height=he, pointsize=pt, family=fo, bg="white")
par(oma=rep(0,4), mar=c(4.2,4.2,0.2,0.2))

if(set.list==0) set.list <- seq(along=Diff$ID) # plot them all

for(i in set.list){
message("Making diagnostic plots for set ", Diff$ID[[i]])

cat("\\subsection{Data set ", Diff$ID[i], "}")

# useful quantities
betac <- Diff$LAT[substring(Diff$CollName[i], 1, 3), "betay"]
sigmac <- Diff$syC[i]
N <- median(Diff$N[[i]], na.rm=TRUE) *1e9
eN <- mad(Diff$N[[i]], na.rm=TRUE) *1e9
W <- median(Res$Wy[[i]], na.rm=TRUE)
eW <- mad(Res$Wy[[i]], na.rm=TRUE)
```

```
ti <- Diff$ti[i] # plot range
tf <- Diff$tf[i]
tl <- c(ti, tf)

# make e-lens diagnostic plot
message("Making e-lens diagnostic plot...")
basename <- paste(Diff$ddir, "/", Diff$ID[i], "/", "tmp_diag_elens", sep="")
pdf(file=paste(basename, ".pdf", sep=""))
dev.control(displaylist="enable") # enable recording
par(oma=rep(0,4), mar=c(3.2,3.2,0.2,3.2), mgp=c(2,0.7,0), tcl=-0.35)
Dev <- list(
  col = c("darkblue", "black", "blue", "magenta"),
  nam = c("T:L2SG", "T:L2SM", "T:L2MCV1", "T:L2COLI"),
  dsc = c("Gun solenoid [kG]", "Main solenoid [kG]",
          "Modulator setting [V]", "Average current [mA]") )
# gun solenoid
t <- Diff$tSG[[i]]
y <- Diff$SG[[i]]
colo <- Dev$col[1]
plot(t, y, xlim=tl, ylim=c(0,4),
     xlab="", ylab="", type="l", col=colo, axes=FALSE)
axis(1)
axis(4, mgp=c(-3, -1.7, 0), tcl=0.35, col=colo, col.axis=colo)
# main solenoid
t <- Diff$tSM[[i]]
y <- Diff$SM[[i]]
colo <- Dev$col[2]
plot.window(xlim=tl, ylim=c(0,30))
points(t, y, type="l", col=colo)
axis(4, col=colo, col.axis=colo)
# modulator voltage setting
t <- Diff$teV[[i]]
y <- Diff$eV[[i]]
colo <- Dev$col[3]
plot.window(xlim=tl, ylim=c(0,800))
points(t, y, type="l", col=colo)
axis(2, mgp=c(-3, -1.7, 0), tcl=0.35, col=colo, col.axis=colo)
# average electron current
t <- Diff$teI[[i]]
```

```
y <- Diff$eI[[i]]
colo <- Dev$col[4]
plot.window(xlim=tl, ylim=c(0, max(80, max(y))))
points(t, y, type="l", col=colo)
axis(2, col=colo, col.axis=colo)
# experiment times
rug(Diff$StepTimes[[i]][,"ts"], quiet=FALSE, col="red")
legend("bottomleft", bty="n", legend="Collimator steps", text.col="red")
# legend
box()
legend("topleft", legend=paste("Data set", Diff$ID[i]))
legend("left", bty="n", legend=Dev$nam, text.col=Dev$col)
mtext("Time [h]", 1, 2)
mtext(Dev$dsc[1], 4, 2, adj=0.25, col=Dev$col[1])
mtext(Dev$dsc[2], 4, 2, adj=0.75, col=Dev$col[2])
mtext(Dev$dsc[3], 2, 2, adj=0.25, col=Dev$col[3])
mtext(Dev$dsc[4], 2, 2, adj=0.75, col=Dev$col[4])
dev.copy2eps(file=paste(basename, ".eps", sep=""))
dev.off()
IncludePlotInLatex(file=basename, flo=FALSE,
                   cap="Electron lens parameters.",
                   lab=paste("fig:elens", Diff$ID[i], sep="_"))

# make beam diagnostic plot
message("Making beam diagnostic plot...")
basename <- paste(Diff$ddir, "/", Diff$ID[i], "/", "tmp_diag_beam", sep="")
pdf(file=paste(basename, ".pdf", sep=""))
dev.control(displaylist="enable") # enable recording
par(oma=rep(0,4), mar=c(3.2,3.2,0.2,3.2), mgp=c(2,0.7,0), tcl=-0.35)
Dev <- list(
  col = c("black", "darkgreen", "darkred", "darkorange"),
  nam = c("T:SBDAIS", "T:SBDAMS", "T:SLASH", "T:SLASV"),
  dsc = c("Beam intensity [1E9]", "Momentum spread [1E-4]",
          "Horizontal beam size [mm]", "Vertical beam size [mm]") )
# beam intensity
t <- Diff$tN[[i]]
y <- Diff$N[[i]]
colo <- Dev$col[1]
plot(t, y, xlim=tl, ylim=range(y)+diff(range(y))*c(-1,0),
     xlab="", ylab="", type="l", col=colo, axes=FALSE)
```

```
axis(1)
axis(4, mgp=c(-3, -1.7, 0), tcl=0.35, col=colo, col.axis=colo)
# momentum spread
t <- Diff$tdpp[[i]]
y <- Diff$dpp[[i]]
colo <- Dev$col[2]
plot.window(xlim=tl, ylim=range(y)+diff(range(y))*c(-1,1))
points(t, y, type="l", col=colo)
axis(4, col=colo, col.axis=colo)
# horizontal beam size
t <- Diff$tSL[[i]]
y <- Diff$sxSL[[i]]
colo <- Dev$col[3]
plot.window(xlim=tl, ylim=range(y)+2*diff(range(y))*c(0,1))
points(t, y, type="l", col=colo)
axis(2, mgp=c(-3, -1.7, 0), tcl=0.35, col=colo, col.axis=colo)
# vertical beam size
t <- Diff$tSL[[i]]
y <- Diff$sySL[[i]]
colo <- Dev$col[4]
plot.window(xlim=tl, ylim=range(y)+2*diff(range(y))*c(0,1))
points(t, y, type="l", col=colo)
axis(2, col=colo, col.axis=colo)
# experiment times
rug(Diff$StepTimes[[i]][,"ts"], quiet=FALSE, col="red")
legend("bottomleft", bty="n", legend="Collimator steps", text.col="red")
# legend
box()
legend("topleft",
       legend=c(paste("Data set ", Diff$ID[i],
         "\nbunches A", Diff$Bu1[i], "-A", Diff$Bu2[i], sep=""))
legend("top", bty="n", legend=Dev$nam, text.col=Dev$col)
mtext("Time [h]", 1, 2)
mtext(Dev$dsc[1], 4, 2, adj=0.25, col=Dev$col[1])
mtext(Dev$dsc[2], 4, 2, adj=0.75, col=Dev$col[2])
mtext(Dev$dsc[3], 2, 2, adj=0.25, col=Dev$col[3])
mtext(Dev$dsc[4], 2, 2, adj=0.75, col=Dev$col[4])
legend("bottomleft", bty="n", legend="Collimator steps", text.col="red")
dev.copy2eps(file=paste(basename, ".eps", sep=""))
dev.off()
```

```
IncludePlotInLatex(file=basename, flo=FALSE,
                   cap="Beam parameters.",
                   lab=paste("fig:beam", Diff$ID[i], sep="_"))

# calibration of losses
message("Plotting loss rates vs. particle losses...")
basename <- paste(Diff$ddir, "/", Diff$ID[i], "/", "tmp_diag_losscal", sep="")
pdf(file=paste(basename, ".pdf", sep=""))
dev.control(displaylist="enable") # enable recording
par(oma=rep(0,4), mar=c(3.2,3.2,0.2,3.2), mgp=c(2,0.7,0), tcl=-0.35)
x <- Res$dNm[[i]]
y <- Res$ILm[[i]]
plot(x, y, xlab="Total number of lost particles",
     ylab="Integrated loss rate")
curve(Res$Kg0[i]+Res$Kg[i]*x, range(x, na.rm=TRUE),
      n=11, col="red", add=TRUE)
legend("topleft", bty="n", legend=Diff$rubber.stamp[i])
legend("bottomright", bty="n",
       legend = paste("Kg = ", formatC(Res$Kg[i], di=4),
                      " +/- ", formatC(Res$eKg[i], di=3), sep=""),
       cex=0.8)
dev.copy2eps(file=paste(basename, ".eps", sep=""))
dev.off()
IncludePlotInLatex(file=basename, flo=FALSE,
                   cap="Calibration of loss rates",
                   lab=paste("fig:losscal", Diff$ID[i], sep="_"))

# overview of collimator scan
message("Plotting scan overview...")
basename <- paste(Diff$ddir, "/", Diff$ID[i], "/", "tmp_diag_cscan", sep="")
pdf(file=paste(basename, ".pdf", sep=""))
dev.control(displaylist="enable") # enable recording
par(oma=rep(0,4), mar=c(3.2,3.2,0.2,3.2), mgp=c(2,0.7,0), tcl=-0.35)
Dev <- list(
  col = c("red", "black"),
  dsc = c("Collimator position [mils]", "Local loss rate [arb. units]") )
# collimator positions
t <- Diff$tCP[[i]]
y <- Diff$CP[[i]]
colo <- Dev$col[1]
```

```
plot(t, y, type="l", xlim=tl, ylim=range(y, finite=TRUE),
     xlab="Time [h]", ylab=Dev$dsc[1], col=Dev$col[1])
# local losses
t <- Diff$tLo[[i]]
y <- Diff$Lo[[i]]
colo <- Dev$col[2]
plot.window(xlim=tl, ylim=range(y, finite=TRUE))
points(t, y, type="l", col=Dev$col[2])
axis(4)
mtext(Dev$dsc[2], 4, 2)
legend("topleft", bty="n", legend=Diff$rubber.stamp[i])
dev.copy2eps(file=paste(basename, ".eps", sep=""))
dev.print(device=png, file=paste(basename, ".png", sep=""), width=wi,
          units="in", res=150)
dev.off()
IncludePlotInLatex(file=paste(basename, ".png", sep=""), flo=FALSE,
                   cap="Overview of collimator scan.",
                   lab=paste("fig:cscan", Diff$ID[i], sep="_"))

# individual collimator steps
message("Plotting individual collimator steps...")
nana <- nrow(Diff$StepTimes[[i]]) # total number of steps
ppp <- 4 # plots per page
nc <- ceiling(sqrt(3*ppp/4)) # number of columns
nr <- ceiling(sqrt(ppp)) # number of rows
npages <- ceiling(nana / ppp) # number of pages
ex1 <- seq(1, nana, by=ppp) # first and last step numbers for each page
ex2 <- pmin(ex1+ppp-1, nana)
for( pn in 1:npages) {
  rangetxt <- paste(formatC(ex1[pn], width=2, flag="0"), "-",
                    formatC(ex2[pn], width=2, flag="0"), sep="")
  basename <- paste(Diff$ddir, "/", Diff$ID[i], "/", "tmp_steps_",
                    rangetxt, sep="")
  pdf(file=paste(basename, ".pdf", sep=""))
  dev.control(displaylist="enable") # enable recording
  par(oma=rep(0,4), mar=c(3.4, 3.4, 0.4, 0.4),
      mgp=c(2,0.5,0), tcl=-0.25)
  layout(matrix(1:ppp, nrow=nr, ncol=nc, byrow=TRUE))
  for(ex in ex1[pn]:ex2[pn]){
      t <- Diff$tLo[[i]]
```

```
sub <- Diff$StepTimes[[i]][ex, "t1"] <= t &
       t <= Diff$StepTimes[[i]][ex, "t2"]
t <- (t[sub]-Diff$StepTimes[[i]][ex, "ts"])*3600
y <- Diff$Lo[[i]][sub]
FittedLossRate <- function(x) {
  LR <- 0*x
  LR <- loss.rate(t=x, K=1, B=Res$Bg[i], t0=Res$t0f[[i]][ex],
            Ai=Res$DAi[[i]][ex]/Res$D[[i]][ex],
            Af=Res$DAf[[i]][ex]/Res$D[[i]][ex],
            Jci=Res$Jci[[i]][ex], Jcf=Res$Jcf[[i]][ex],
            D=Res$D[[i]][ex], dt=Res$dtf[[i]][ex])
  return(LR)}
t.peak <- seq(-0.2, 2, by=0.04)
y.peak <- FittedLossRate(t.peak)
yl <- range(c(y, y.peak), na.rm=TRUE)
dyl <- diff(yl)
yl[2] <- yl[2]+dyl/2
plot(t, y, col=rgb(0,0,1, 0.2), pch=20, cex=0.4, xlab="", ylab="",
     ylim=yl)
tm <- c(t.peak,
        seq(min(t, na.rm=TRUE), max(t, na.rm=TRUE), length=64))
tm <- sort(tm)
ym <- FittedLossRate(tm)
points(tm, ym, type="l", col="magenta")
legend("topleft", bty="n", cex=0.8,
  legend = c(
    paste("Step #", ex),
    paste("ts = ", formatC(Diff$StepTimes[[i]][ex, "ts"], di=7), " h",
          sep=""),
    paste("CP = ", Res$cpbef[[i]][ex], " to ", Res$cpaft[[i]][ex],
          " mils", sep=""),
    paste("y = ",
          formatC(5e-4*abs(Res$xci[[i]][ex]+Res$xcf[[i]][ex]),
                  di=3), " mm", sep=""),
    paste("(", formatC(abs(Res$xsi[[i]][ex]), di=3), " sigma)", sep=""),
    paste("J = ",
          formatC(0.5*(Res$Jci[[i]][ex]+Res$Jcf[[i]][ex]),
                  di=3), " um", sep="") ))
legend("topright", bty="n", cex=0.8,
       legend=c(
```

```
                paste("Dg.pk = ", formatC(Res$Dg.pk[[i]][ex], di=3, format="e")),
                paste("Dg.tr = ", formatC(Res$Dg.tr[[i]][ex], di=3, format="e"),
                      "+/-", formatC(Res$eDg.tr[[i]][ex], di=2, format="e")),
                paste("D =", formatC(Res$D[[i]][ex], di=3, format="e"),
                      "+/-", formatC(Res$eD[[i]][ex], di=2, format="e")),
                paste("DAi =", formatC(Res$DAi[[i]][ex], di=4),
                      "+/-", formatC(Res$eDAi[[i]][ex], di=2)),
                paste("DAf =", formatC(Res$DAf[[i]][ex], di=4),
                      "+/-", formatC(Res$eDAf[[i]][ex], di=2)),
#               paste("t0 =", formatC(Res$t0f[[i]][ex], di=3),
#                      "+/-", formatC(Res$et0f[[i]][ex], di=2)),
#               paste("dt =", formatC(Res$dtf[[i]][ex], di=3),
#                      "+/-", formatC(Res$edtf[[i]][ex], di=2)),
                paste("Chi2 / dof =", formatC(Res$chi2[[i]][ex], di=3), "/",
                      Res$dof[[i]][ex], "=",
                      formatC(Res$chi2[[i]][ex]/Res$dof[[i]][ex], di=3)),
                paste("conv =", Res$conv[[i]][ex])) )
  }
  layout(1)
  mtext("Time [s]", side=1, line=2)
  mtext("Loss rate [arb. units]", side=2, line=2)
  dev.copy2eps(file=paste(basename, ".eps", sep=""))
  dev.off()
  IncludePlotInLatex(file=basename, flo=FALSE,
                     cap=paste("Collimator steps ", rangetxt, ".", sep=""),
                     lab=paste("fig:csteps", Diff$ID[i], rangetxt, sep="_"))
}

# find steps with valid results
j <- which(!is.na(Res$D[[i]]))
if(length(j>0)){

# horizontal axes and auxiliary quantities
x <- abs(0.5*(Res$xci[[i]][j]+Res$xcf[[i]][j])/1e3) # [mm]
xs <- x / sigmac # [sigmas]
Ji <- Res$Jci[[i]][j]
Jf <- Res$Jcf[[i]][j]
J <- 0.5*(Ji+Jf)
dJ <- abs(Ji-Jf)
sigma.labels <- seq(from=0,
```

```
                    to=ceiling(max(xs, na.rm=TRUE)))
if(length(sigma.labels)<2) sigma.labels <- 0:10
sigma.labels.pos <- sigma.labels*sigmac
J.labels <- seq(0, ceiling(max(J)), by=0.05)
if(length(J.labels)<2) J.labels <- seq(0, 0.2, by=0.1)
J.labels.pos <- sqrt( J.labels * betac )

# plot symbols
sym <- 1+15*Res$stepin[[i]][j]

# diffusion vs. amplitude
message("Plotting diffusion vs. amplitude...")
basename <- paste(Diff$ddir, "/", Diff$ID[i], "/", "tmp_diff_coeff", sep="")
pdf(file=paste(basename, ".pdf", sep=""))
dev.control(displaylist="enable") # enable recording
par(oma=rep(0,4), mar=c(3.2,3.2,3.2,0.2), mgp=c(2,0.5,0), tcl=-0.25)
x0 <- c(-0.02, 0.02) # artificially displaced for clarity
y0 <- c(Res$D0x[i], Res$D0y[i])
ey0 <- c(Res$eD0x[i], Res$eD0y[i])
X <- c(x, x0)
xl <- range(X)
y <- Res$D[[i]][j]
ey <- Res$eD[[i]][j]
ypk <- Res$Dg.pk[[i]][j]
ytr <- Res$Dg.tr[[i]][j]
eytr <- Res$eDg.tr[[i]][j]
yg <- Res$Dg[[i]][j]
Y <- c(y0, y, ypk, ytr, yg)
yl <- range(Y[Y>0 & is.finite(Y)], na.rm=TRUE)
plot(x, ypk, xlim=xl, ylim=yl, log="y",
  xlab=expression(paste("Vertical collimator position, ", y[c], " [mm]",
                        sep="")),
  ylab=expression(paste("Diffusion rate, D [", mu*m^2/s, "]", sep="")),
  pch=sym, col="cyan")
points(x, ytr, pch=sym, col="darkblue")
arrows(x, ytr-eytr, x, y+eytr, le=0.05, angle=90, code=3, col="darkblue")
points(x, yg, pch=sym, col="green")
points(x, y, pch=sym, col="magenta")
arrows(x, y-ey, x, y+ey, le=0.05, angle=90, code=3, col="magenta")
points(x0, y0, pch=c(0, 15))
```

```
arrows(x0, y0-ey0, x0, y0+ey0, le=.05, angle=90, code=3)
axis(3, at=sigma.labels.pos, labels=sigma.labels)
mtext(expression(paste("Vertical collimator position, ", y[c], " [",
    sigma, "]", sep="")), line=2, side=3)
legend("topleft", bty="n", legend=Diff$rubber.stamp[i])
legend("bottomright", bty="n", text.col=c("magenta", "cyan",
                                "darkblue", "green"),
       legend=c("Fit results", "Estimates from peak",
       "Estimates from transient", "Estimates from 1-par fit"))
dev.copy2eps(file=paste(basename, ".eps", sep=""))
dev.off()
IncludePlotInLatex(file=basename, flo=FALSE,
                   cap="Diffusion vs. amplitude.",
                   lab=paste("fig:DvsJ", Diff$ID[i], sep="_"))

k <- Res$ILlost[[i]][j] / Res$Nlost[[i]][j]
ek <- 0.5*abs( Res$ILlost[[i]] / (Res$Nlost[[i]]-Res$eNlost[[i]]) -
               Res$ILlost[[i]] / (Res$Nlost[[i]]+Res$eNlost[[i]])   )[j]
Kg <- Res$Kg[i]

# calibration factor vs. amplitude
message("Plotting loss calibration factor vs. amplitude...")
basename <- paste(Diff$ddir, "/", Diff$ID[i], "/", "tmp_diag_calvsampl", sep="")
pdf(file=paste(basename, ".pdf", sep=""))
dev.control(displaylist="enable") # enable recording
par(oma=rep(0,4), mar=c(3.2,3.2,3.2,0.2), mgp=c(2,0.7,0), tcl=-0.35)
y <- k
ey <- ek
plot(c(0, x), c(Kg, y), log="y", type="n",
     xlab=expression(paste("Vertical collimator position, ", y[c], " [mm]",
                           sep="")),
     ylab="Calibration factor, k [(loss rate units) / (particles/s)]",
     pch=sym)
abline(h=Kg, col="gray"); box()
points(x, y, pch=sym)
arrows(x, y-ey, x, y+ey, le=.05, angle=90, code=3)
axis(3, at=sigma.labels.pos, labels=sigma.labels)
mtext(expression(paste("Vertical collimator position, ", y[c], " [",
    sigma, "]", sep="")), line=2, side=3)
legend("topright", bty="n", legend=Diff$rubber.stamp[i])
```

```
legend("bottomleft", bty="n",
       legend = paste("Kg = ", formatC(Kg, di=4)),
       cex=0.8)
dev.copy2eps(file=paste(basename, ".eps", sep=""))
dev.off()
IncludePlotInLatex(file=basename, flo=FALSE,
                   cap="Loss-rate calibration factor vs. amplitude",
                   lab=paste("fig:calvsampl", Diff$ID[i], sep="_"))

# steady-state losses
message("Plotting steady-state losses...")
basename <- paste(Diff$ddir, "/", Diff$ID[i], "/", "tmp_steady_state", sep="")
pdf(file=paste(basename, ".pdf", sep=""))
dev.control(displaylist="enable") # enable recording
par(oma=rep(0,4), mar=c(3.2,3.2,3.2,0.2), mgp=c(2,0.5,0), tcl=-0.25)
yi <- Res$DAi[[i]][j]
eyi <- Res$eDAi[[i]][j]
yf <- Res$DAf[[i]][j]
eyf <- Res$eDAf[[i]][j]
Y <- c(yi, yf)
yl <- range(Y[Y>0 & is.finite(Y)], na.rm=TRUE)
coli <- "lightgreen"
colf <- "darkgreen"
plot(x, yi, xlim=range(c(0, x)), ylim=yl, log="y",
  xlab=expression(paste("Vertical collimator position, ", y[c], " [mm]",
                        sep="")),
  ylab=expression(paste("Steady-state loss rate, ", k*D*A,
      " [loss monitor units]", sep="")),
  pch=sym, col=coli)
arrows(x, yi-eyi, x, yi+eyi, le=0.05, angle=90, code=3, col=coli)
points(x, yf, pch=sym, col=colf)
arrows(x, yf-eyf, x, yf+eyf, le=0.05, angle=90, code=3, col=colf)
axis(3, at=sigma.labels.pos, labels=sigma.labels)
mtext(expression(paste("Vertical collimator position, ", y[c], " [",
    sigma, "]", sep="")), line=2, side=3)
legend("topright", bty="n", legend=Diff$rubber.stamp[i])
legend("bottomright", bty="n", text.col=c(coli, colf),
       legend=c("Initial (before step)", "Final (after step)"))
dev.copy2eps(file=paste(basename, ".eps", sep=""))
dev.off()
```

```
IncludePlotInLatex(file=basename, flo=FALSE,
                   cap="Steady-state loss rate vs. amplitude.",
                   lab=paste("fig:kDAvsJ", Diff$ID[i], sep="_"))

# particle flux vs amplitude
message("Plotting particle flux vs amplitude...")
basename <- paste(Diff$ddir, "/", Diff$ID[i], "/", "tmp_flux", sep="")
pdf(file=paste(basename, ".pdf", sep=""))
dev.control(displaylist="enable") # enable recording
par(oma=rep(0,4), mar=c(3.2,3.2,3.2,0.2), mgp=c(2,0.5,0), tcl=-0.25)
x0 <- 0
X <- c(x, x0)
y <- Res$DAi[[i]][j] / k
ey <- y * sqrt( (Res$eDAi[[i]][j]/Res$DAi[[i]][j])^2 + (ek/k)^2 )
phi0 <- Res$lambda[i] * median(Diff$N[[i]]) * 1e9
ephi0 <- Res$elambda[i] * median(Diff$N[[i]]) * 1e9
Y <- c(y, phi0)
EY <- c(ey, ephi0)
plot(X, Y, log="y",
  xlab=expression(paste("Vertical collimator position, ", y[c], " [mm]",
                        sep="")),
  ylab=expression(paste("Particle flux, ", phi = D*A,
      " [particles/s]", sep="")),
  pch=c(sym, 15))
arrows(X, Y-EY, X, Y+EY, le=0.05, angle=90, code=3)
axis(3, at=sigma.labels.pos, labels=sigma.labels)
mtext(expression(paste("Vertical collimator position, ", y[c], " [",
    sigma, "]", sep="")), line=2, side=3)
legend("topleft", bty="n", legend=Diff$rubber.stamp[i])
legend("topright", bty="n",
       legend = c(
         "Intensity decay rate:",
         paste("  lambda =", formatC(Res$lambda[i], di=4), "+/-",
               formatC(Res$elambda[i], di=3), "1/s"),
         paste("  (tau =", formatC(1/Res$lambda[i]/3600, di=4), "h)"),
         "Emittance growth rates:",
         paste("  gamma_x =", formatC(Res$gammax[i], di=4), "+/-",
               formatC(Res$egammax[i], di=3), "1/s"),
         paste("  gamma_y =", formatC(Res$gammay[i], di=4), "+/-",
               formatC(Res$egammay[i], di=3), "1/s")
```

```
       ),
       cex=0.8)
dev.copy2eps(file=paste(basename, ".eps", sep=""))
dev.off()
IncludePlotInLatex(file=basename, flo=FALSE,
                   cap="Particle flux vs. amplitude.",
                   lab=paste("fig:DAvsJ", Diff$ID[i], sep="_"))

# halo population gradient
message("Plotting halo population gradient...")
basename <- paste(Diff$ddir, "/", Diff$ID[i], "/", "tmp_halo_grad", sep="")
pdf(file=paste(basename, ".pdf", sep=""))
dev.control(displaylist="enable") # enable recording
par(oma=rep(0,4), mar=c(3.2,3.2,3.2,0.2), mgp=c(2,0.5,0), tcl=-0.25)
x0 <- 0
y0 <- 0.5 * N / W^2
ey0 <- 0.5 * sqrt( (eN/N)^2 + (2*eW/W)^2 )
y <- Res$DAi[[i]][j] / k / Res$D[[i]][j]
ey <- y * sqrt( (Res$eDAi[[i]][j]/Res$DAi[[i]][j])^2 + (ek/k)^2 +
                (Res$eD[[i]][j]/Res$D[[i]][j])^2 )
X <- c(x0, x)
Y <- c(y0, y)
EY <- c(ey0, ey)
plot(X, Y, log="y", type="n",
  xlab=expression(paste("Vertical collimator position, ", y[c], " [mm]",
                        sep="")),
  ylab=expression(paste("Halo population gradient, A [particles/", mu*m^2, "]", sep="")),
  pch=c(15, sym))
xscore <- seq(0, 8, by=0.1)
xcore <- xscore * sigmac
Jcore <- xcore^2/betac
yGau <- 0.5*N*exp(-Jcore/W)/W^2 # calculated gradient of Gaussian core
points(xcore, yGau, type="l", col="gray")
points(X, Y, pch=c(15, sym))
arrows(X, Y-EY, X, Y+EY, le=0.05, angle=90, code=3)
axis(3, at=sigma.labels.pos, labels=sigma.labels)
mtext(expression(paste("Vertical collimator position, ", y[c], " [",
    sigma, "]", sep="")), line=2, side=3)
legend("topright", bty="n", legend=Diff$rubber.stamp[i])
dev.copy2eps(file=paste(basename, ".eps", sep=""))
```

```
dev.off()
IncludePlotInLatex(file=basename, flo=FALSE,
                   cap="Halo population gradient.",
                   lab=paste("fig:halograd", Diff$ID[i], sep="_"))

# halo population density - what does it mean?
if(0){
message("Plotting halo population density...")
basename <- paste(Diff$ddir, "/", Diff$ID[i], "/", "tmp_halo_dens", sep="")
pdf(file=paste(basename, ".pdf", sep=""))
dev.control(displaylist="enable") # enable recording
par(oma=rep(0,4), mar=c(3.2,3.2,3.2,0.2), mgp=c(2,0.5,0), tcl=-0.25)
x0 <- 0
y0 <- 0.5 * N / W
ey0 <- 0.5 * sqrt( (eN/N)^2 + (eW/W)^2 )
yy <- ifelse(Res$stepin[[i]][j], Res$DAi[[i]][j], Res$DAf[[i]][j])
eyy <- ifelse(Res$stepin[[i]][j], Res$eDAi[[i]][j], Res$eDAf[[i]][j])
y <-  yy / k / Res$D[[i]][j] * dJ
ey <- y * sqrt( (eyy/yy)^2 +
                (ek/k)^2 +
                (Res$eD[[i]][j]/Res$D[[i]][j])^2 )
X <- c(x0, x)
Y <- c(y0, y)
EY <- c(ey0, ey)
plot(X, Y, log="y", type="n",
  xlab=expression(paste("Vertical collimator position, ", y[c], " [mm]",
                        sep="")),
  ylab=expression(paste("Halo population density, ",
      A*(Delta*J), " [particles/", mu*m, "]", sep="")),
  pch=c(15, sym))
# calculated Gaussian core
xscore <- seq(0, 8, by=0.1)
xcore <- xscore * sigmac
Jcore <- xcore^2/betac
yGau <- 0.5*N*exp(-Jcore/W)/W
points(xcore, yGau, type="l", col="gray")
# data
points(X, Y, pch=c(15, sym))
arrows(X, Y-EY, X, Y+EY, le=0.05, angle=90, code=3)
axis(3, at=sigma.labels.pos, labels=sigma.labels)
```

```
axis(3, mgp=c(-3, -1.7, 0), tcl=0.35, at=J.labels.pos, labels=J.labels)
mtext(expression(paste("Vertical collimator position, ", y[c], " [",
    sigma, "]", sep="")), line=2, side=3)
mtext(expression(paste("Vertical collimator action, ", J[c], " [",
    mu*m, "]", sep="")), line=-2, side=3)
legend("topright", bty="n", legend=Diff$rubber.stamp[i])
dev.copy2eps(file=paste(basename, ".eps", sep=""))
dev.off()
IncludePlotInLatex(file=basename, flo=FALSE,
                   cap="Halo population density.",
                   lab=paste("fig:halodens", Diff$ID[i], sep="_"))
}

} # end of plotting of existing data
} # end loop over data sets
} # end of function

# plot analyzed data sets or, if reading from file,
# all data sets that have both data and results
MakeDiagnosticPlots(set.list=8) # test
if(0){
if(is.numeric(RESULTS.SOURCE)){
  MakeDiagnosticPlots(set.list=RESULTS.SOURCE)
} else {
  for(i in 1:min(length(Diff$tSG), length(Res$D))) {
    if(is.numeric(Diff$tSG[[i]]) & is.numeric(Res$D[[i]]))
       MakeDiagnosticPlots(set.list=i) }}
}


Stangle("diana.Rnw", annotate=FALSE) # make R file for inclusion


Cleanup()
```

[1] A. J. Lichtenberg and M. A. Lieberman, *Regular and Chaotic Dynamics* (Springer-Verlag, New York, 1992), p. 320.

[2] T. Chen et al., Phys. Rev. Lett. **68**, 33 (1992).

[3] A. Gerasimov, Report No. FERMILAB-PUB-92-185 (1992).

[4] F. Zimmermann, Part. Accel. **49**, 67 (1995); Report No. SLAC-PUB-6634 (October 1994).

[5] T. Sen and J. A. Ellison, Phys. Rev. Lett. **77**, 1051 (1996).

[6] N. Mokhov et al., to be published in JINST (September 2011); Report No. FERMILAB-PUB-11-378-APC.

[7] L. Burnod, G. Ferioli, and J. B. Jeanneret, Report No. CERN-SL-90-01 (1990).

[8] R. P. Fliller III et al., in *Proc. 2003 Part. Accel. Conf. (PAC03)*, p. 2904 (IEEE, Piscataway, NJ, 2003).

[9] G. Stancari et al., Phys. Rev. Lett. **107**, 084802 (2011), arXiv:1105.3256 [physics.acc-ph].

[10] G. Stancari, Report No. FERMILAB-FN-0926-APC, arXiv:1108.5010 [physics.acc-ph].

[11] K.-H. Mess and M. Seidel, Nucl. Instr. Meth. Phys. Res. A **351**, 279 (1994); M. Seidel, PhD Thesis, Hamburg University, DESY 94-103 (June 1994).

[12] R Development Core Team, *R: A language and environment for statistical computing* (R Foundation for Statistical Computing, Vienna, Austria, 2010), ISBN 3-900051-07-0, R-project.org.

[13] Friedrich Leisch, Sweave: Dynamic generation of statistical reports using literate data analysis, in Compstat 2002 — Proceedings in Computational Statistics (Wolfgang Härdle and Bernd Rönz, editors), pp. 575–580 (Physica Verlag, Heidelberg, 2002), ISBN 3-7908-1517-9.