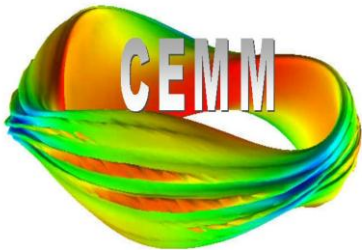


# A method for reconstructing equilibrium energetic particle distributions from NUBEAM

Josh Breslau and Guoyong Fu



CEMM Meeting  
May 1, 2011  
Austin, TX



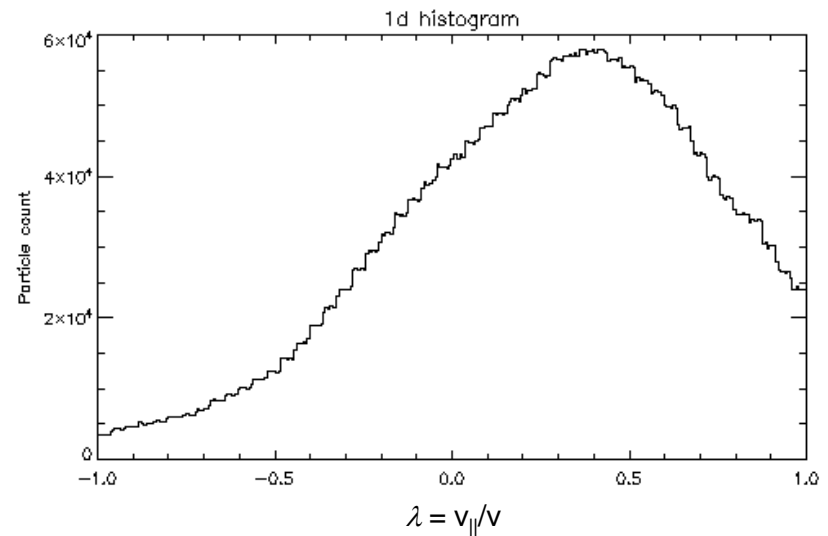
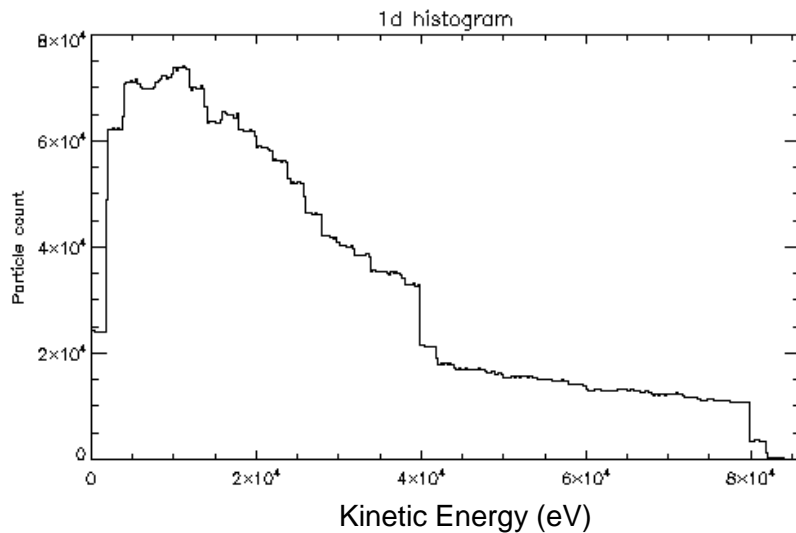
# Outline

- A. NUBEAM output
- B. M3DK, NOVA-K needs
- C. Workflow
  1. Random sampling
  2. Coordinate conversion
  3. Construction of Jacobian
  4. Orbit classification
  5. Binning
  6. Smoothing
  7. Splining
- D. Examples of spline reconstruction
- E. Future Work

# NUBEAM

- Monte Carlo code for 2D time dependent simulation of fast ion species.
- Handles energetic ions from beam injection or fusion reactions.
- Includes guiding center drift orbiting, collisional, and atomic physics effects.
- Works in tandem with transport codes evolving axisymmetric MHD equilibria.

*E.g.*, coupling to TSC through the SWIM Integrated Plasma Simulator for NSTX shot 124379, collecting particle data into 4D bins for output:

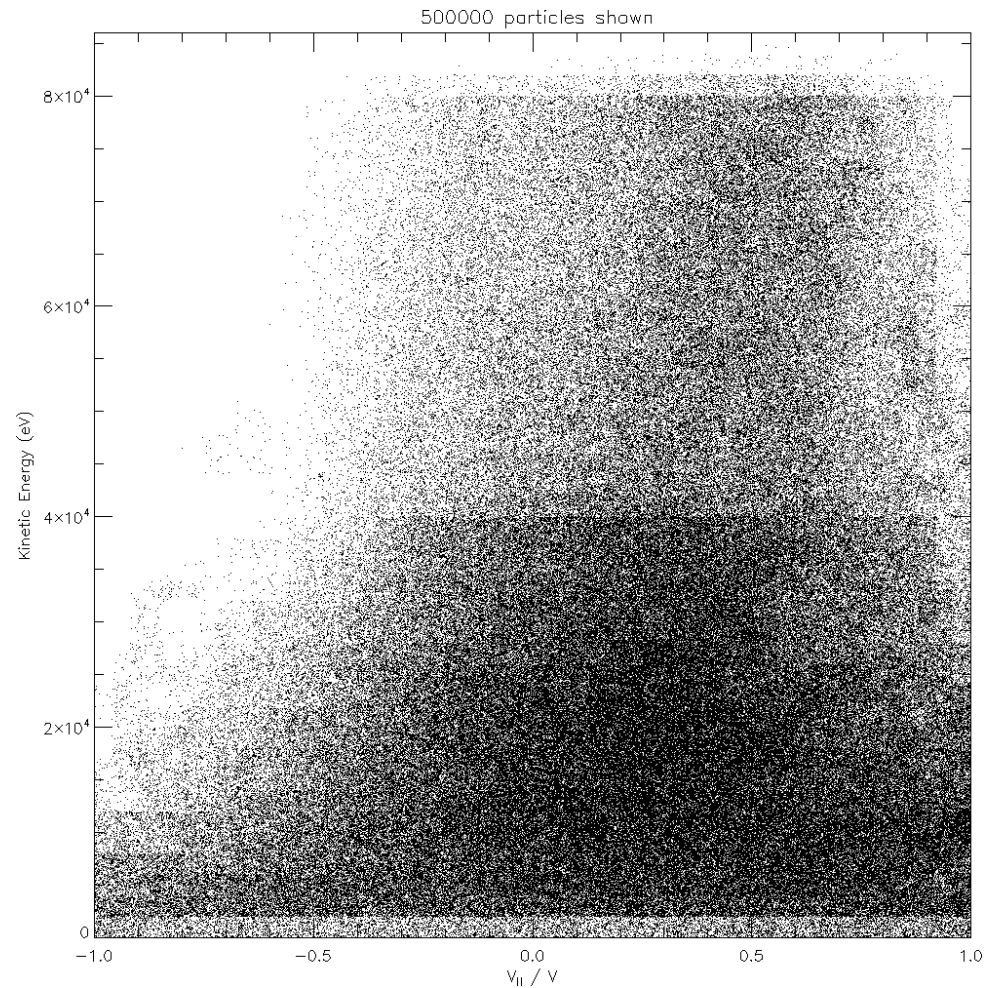
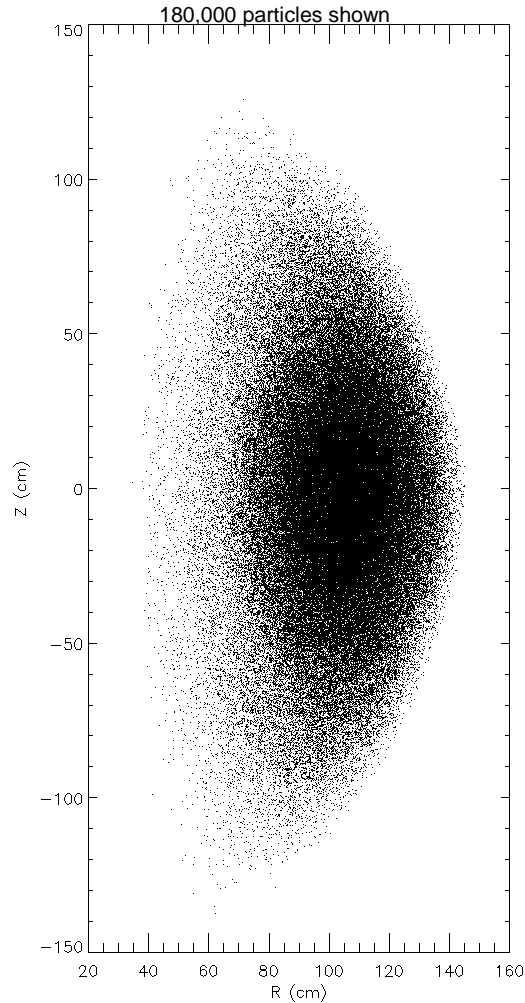


# Kinetic Codes

- NOVA-K computes the modification to ideal linear MHD growth rate due to the energetic particle population.
- M3D-K combines a gyrokinetic  $\delta f$  energetic particle advance with an extended MHD fluid advance for the bulk plasma in nonlinear time-dependent hybrid simulations.
- Both codes are initialized with analytic particle distributions with a small number of free parameters. Initialization from more realistic distributions computed by NUBEAM could improve the accuracy of their calculations.
- These kinetic codes need the distribution function in terms of three constants of the motion, with the fourth dimension eliminated. The function must be smooth enough to allow derivatives to be taken with respect to energy and toroidal angular momentum.

# get\_fbm: NTCC Utility for sampling 4D distribution function

`% get_fbm @cdf_read.ind` (sample size = 10,000,000)



# Transform to 3D Coordinates

$$F(R, z, \lambda, E) \rightarrow f(P_\varphi, \mu, E)$$

Convert (R,z) to meters, E to Joules.

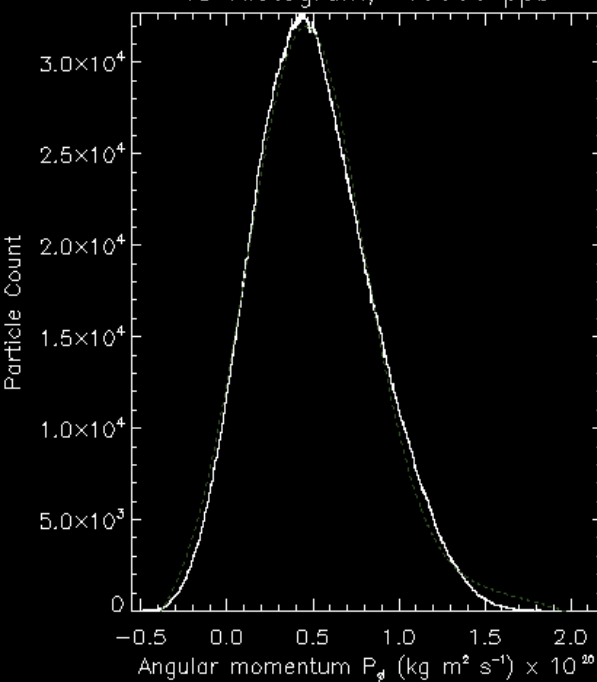
Then transform using

$$P_\varphi \equiv (eA_\varphi + mv_\varphi)R \approx e\psi(R, z) + mv_\parallel R \frac{B_\varphi(R, z)}{B(R, z)}$$

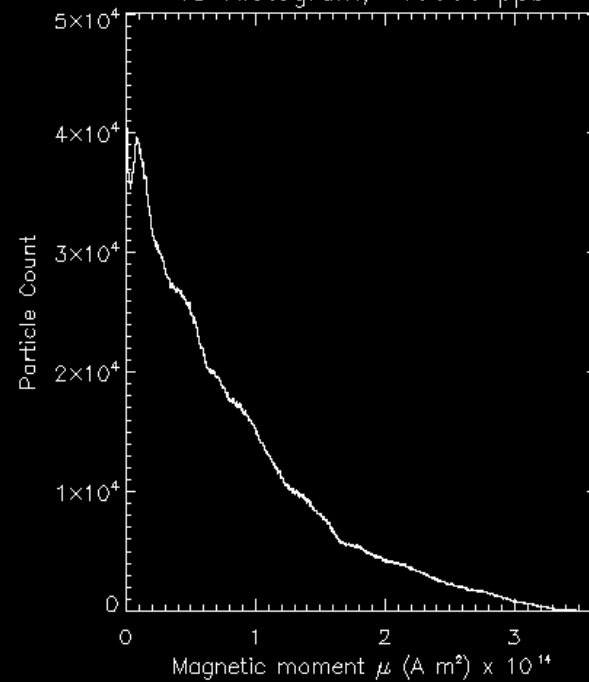
$$\mu = \frac{\frac{1}{2}mv_\perp^2}{B} = \frac{\left[1 - \left(\frac{v_\parallel}{v}\right)^2\right]E}{B(R, z)}$$

Utility part2dist reads particle data, IPS plasma state, uses plasma state interpolation routines to get  $\psi$ ,  $B$ , performs transform.

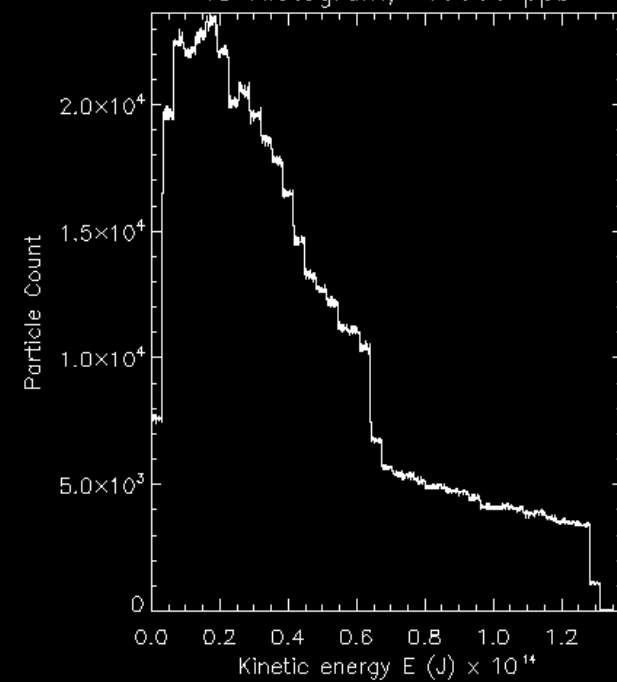
1D Histogram, 10000 ppb



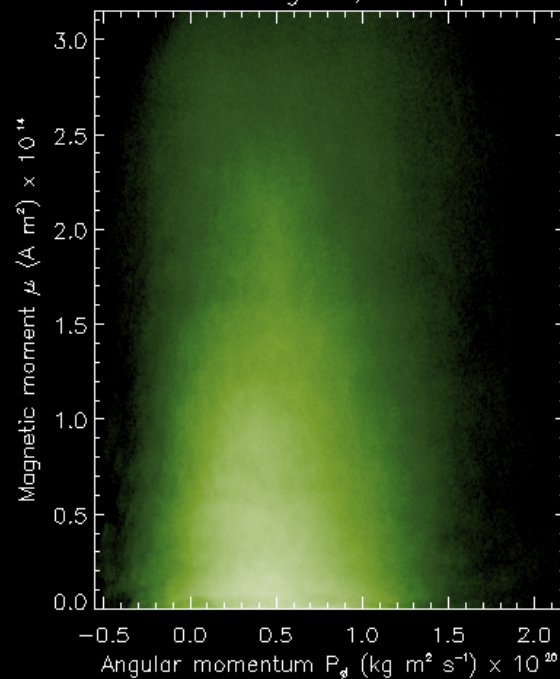
1D Histogram, 10000 ppb



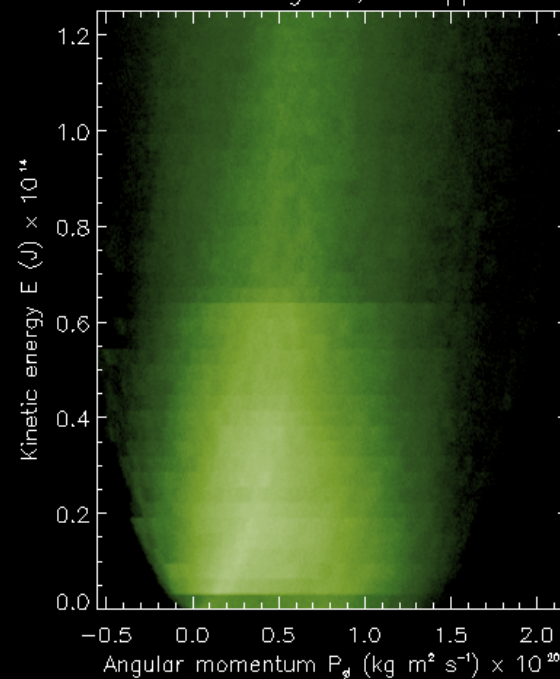
1D Histogram, 10000 ppb



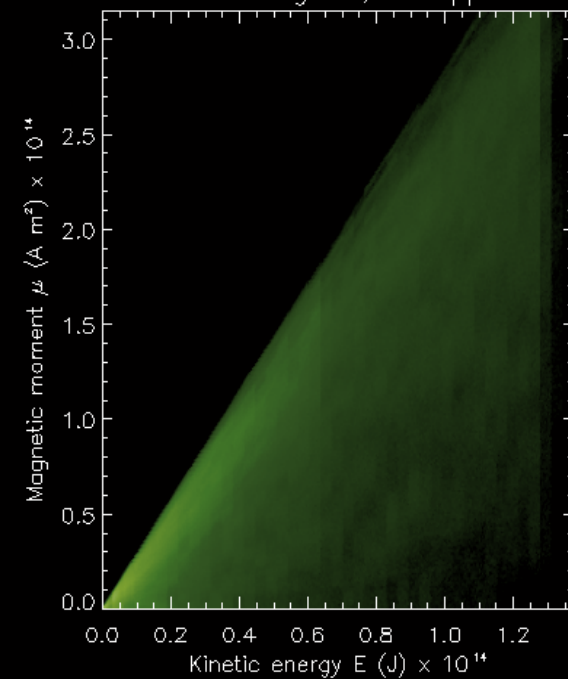
2D Histogram, 100 ppb



2D Histogram, 100 ppb



2D Histogram, 100 ppb



# Jacobian of the Transform

Over any region of phase space,

$$\iiint \iiint F(x, y, z, v_x, v_y, v_z) d^3x d^3v = \iiint f(P_\phi, \mu, E) \mathfrak{J}(P_\phi, \mu, E) dP_\phi d\mu dE$$

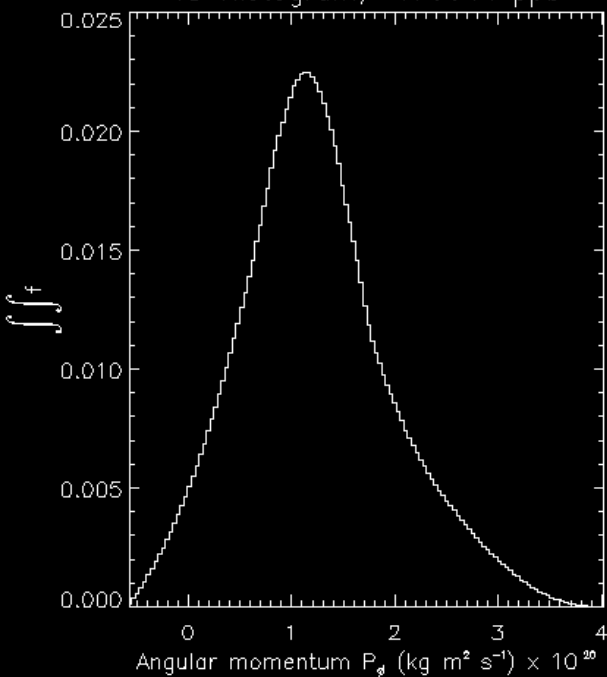
An exact analytic form for  $\mathfrak{J}$  cannot be derived simply as it requires integration over all phase space orbits. A Monte Carlo approach is used, based on the observation that the transform of a uniform distribution in Euclidean phase space is proportional to the inverse of the Jacobian in the new space.

Procedure:

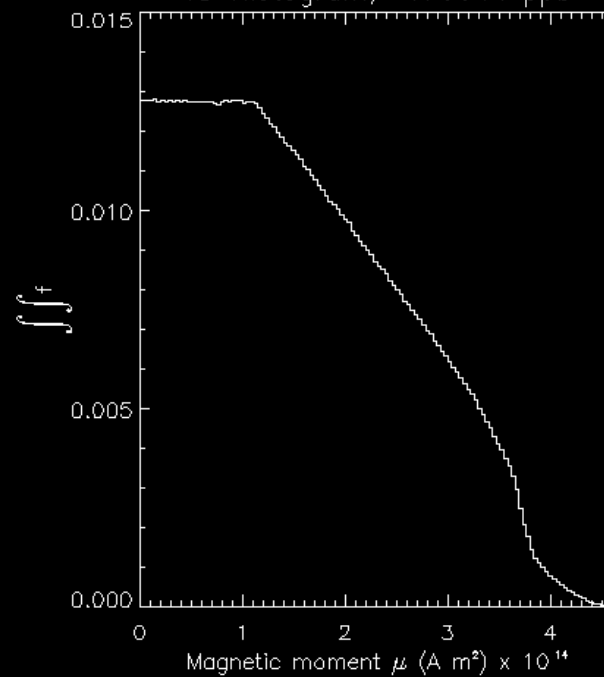
- Generate a population of ~20 million particles in phase space such that  $R^2$ ,  $z$ ,  $v_\perp^2$ , and  $v_\parallel$  are uniform random deviates, within the same range as the NUBEAM distribution.
- Transform coordinates to constants-of-motion space.
- Bin, smooth.



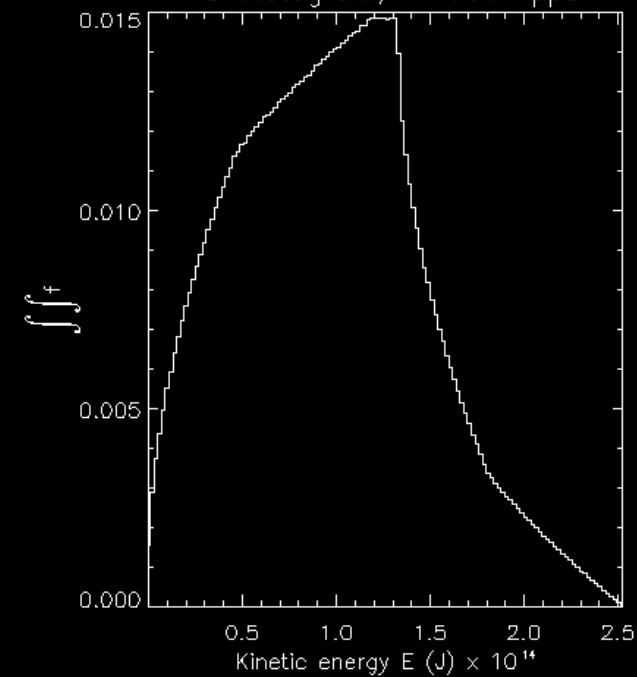
1D Histogram, 179044 ppb



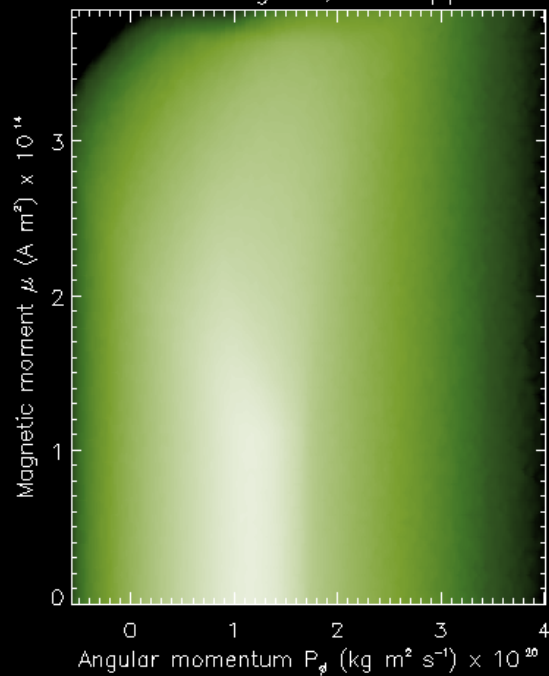
1D Histogram, 179044 ppb



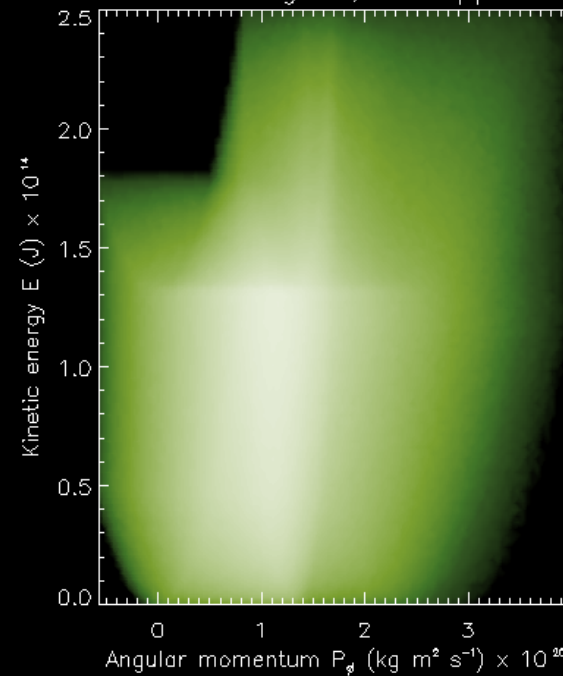
1D Histogram, 179044 ppb



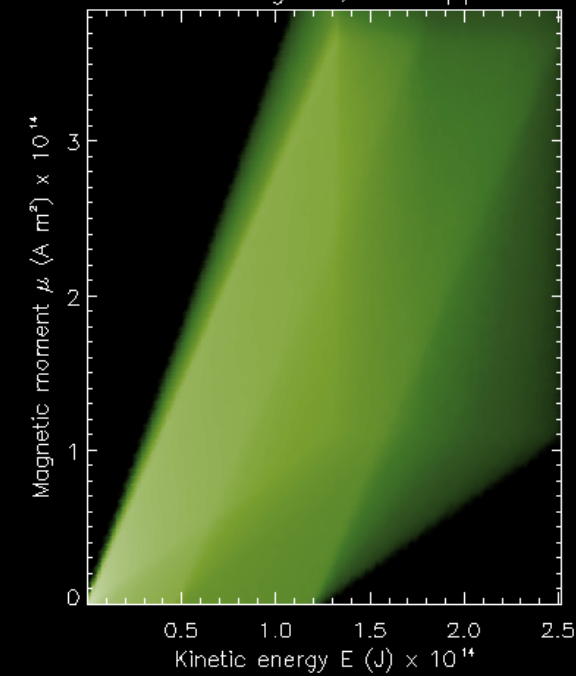
2D Histogram, 2486 ppb



2D Histogram, 2486 ppb







2D Histogram, 2486 ppb

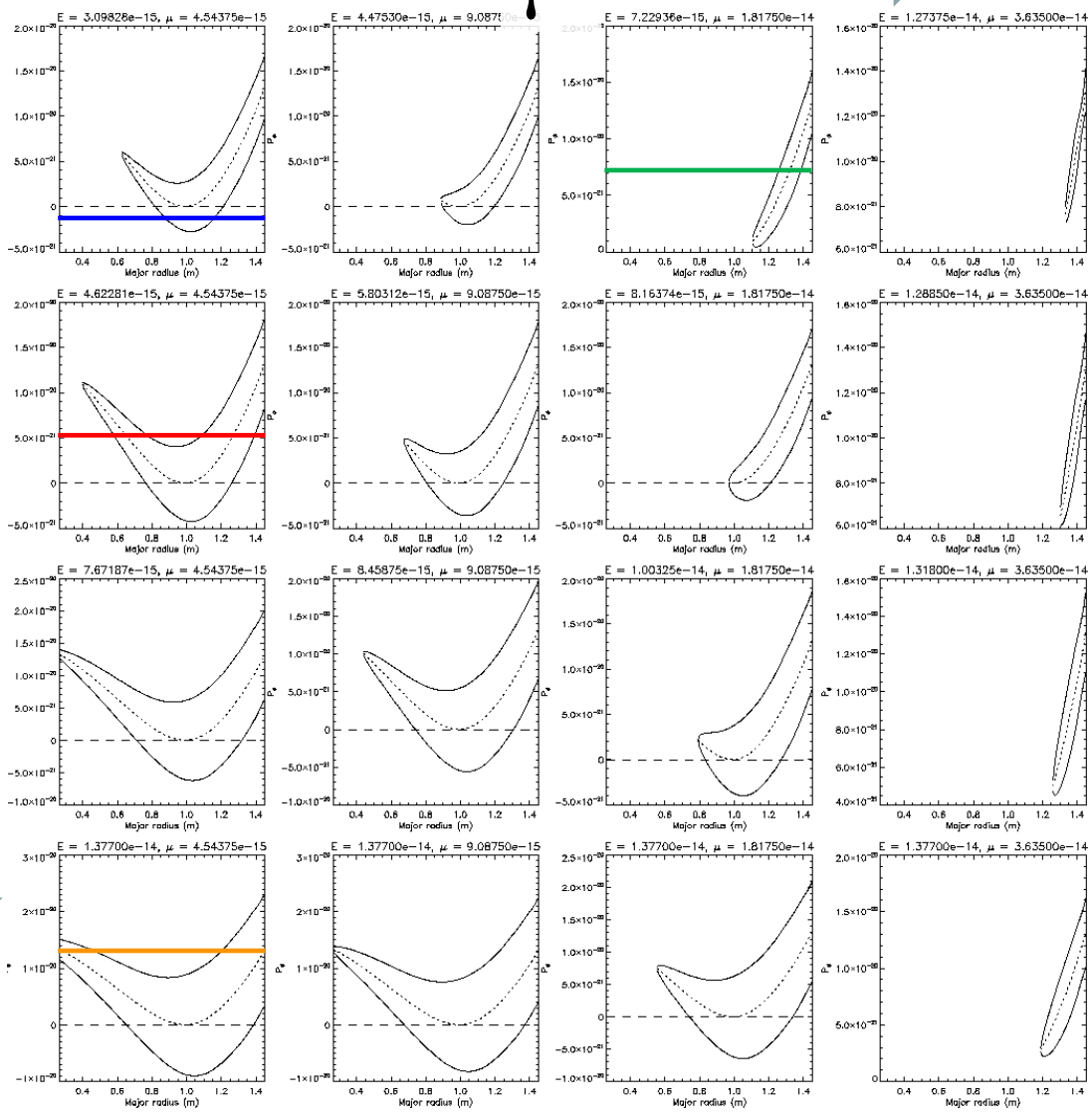


# Ambiguity in $P_\phi$ : $v_{||} = \pm \sqrt{\frac{E - \mu B}{m/2}}$



## Cases:

-  Only the negative root is consistent with the constants of motion: particle is co-passing.
-  Only the positive root is consistent with the constants of motion: particle is counter-passing.
-  Positive and negative roots on outboard side are both consistent with the constants of motion: particle is trapped (in a banana orbit).
-  Multiple solutions are consistent with the constants of motion: additional information on the sign of  $v_{||}$  is needed to resolve orbit type.



All plots are along midplane.

# Constructing $f(P_\phi, \mu, E)$ with `fitEjac`

- Divide the 10,000,000 particles into subsets according to orbit type and sort each subset by magnetic moment. Divide them into several subpopulations of either equal width in  $\mu$  or equal particle count.
- Sort the particles in each subpopulation into a number of bins in the  $P_\phi$  and  $E$  directions, chosen to preserve information but minimize noise.
- Remove lost particles, defined as bins containing a single particle with neighbors containing none.
- Apply Gaussian smoothing in both directions.
- Multiply by smoothed, binned Jacobian.
- Fit 2D cubic B-splines to the smoothed data using `gsl` routines, with uniform knots and a number of coefficients in each direction approximately 5/8 the number of bins.
- Store the spline coefficients in a file. Utility routines have been developed to read the spline data file and perform quick spline and derivative interpolations at arbitrary points in the domain. (Linear interpolation is performed in  $\mu$ ).

# Basis Splines

Given a nondecreasing knot vector

$$\vec{t} = \{t_0, t_1, \dots, t_{n+k-1}\}$$

there are  $n$  basis splines of order  $k$  on the interval, defined recursively by

$$B_{i,1}(x) = \begin{cases} 1, & t_i \leq x < t_{i+1} \\ 0, & \text{otherwise} \end{cases}$$

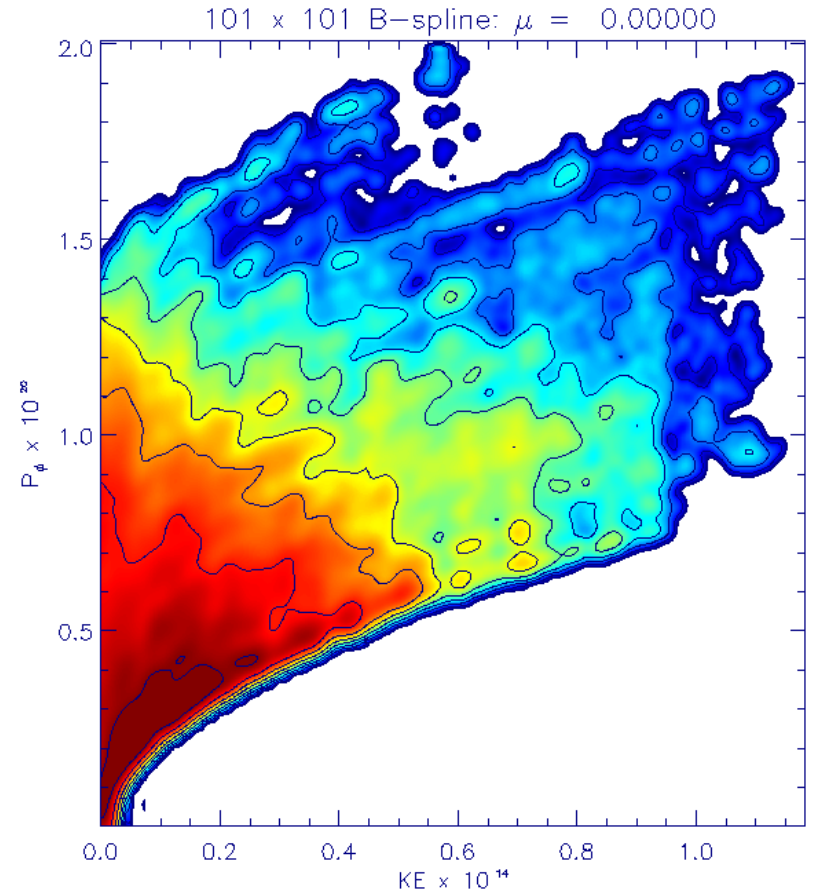
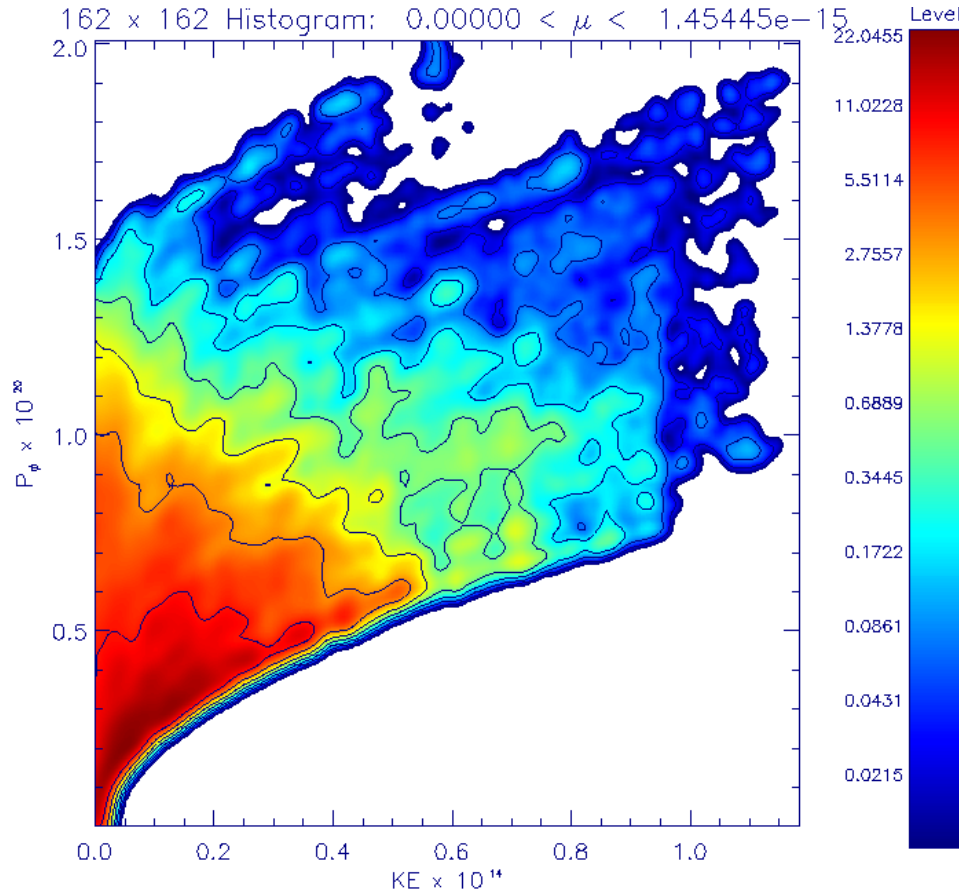
$$B_{i,k}(x) = \frac{x - t_i}{t_{i+k-1} - t_i} B_{i,k-1}(x) + \frac{t_{i+k} - x}{t_{i+k} - t_{i+1}} B_{i+1,k-1}(x)$$

for  $i=0, \dots, n-1$ . For cubic B-splines,  $k=4$ . For an arbitrary function represented at at least  $n+1$  discrete data points, linear least squares fitting can be used to solve for the  $c_i$  in

$$f(x) = \sum_{i=0}^{n-1} c_i B_{i,4}(x).$$

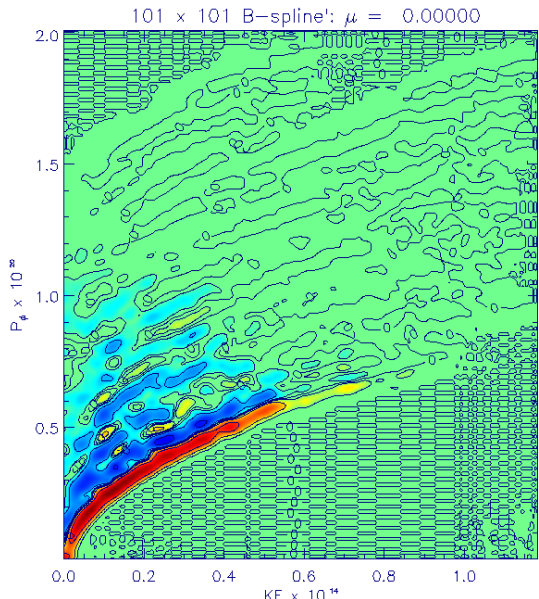
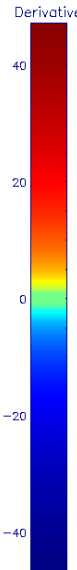
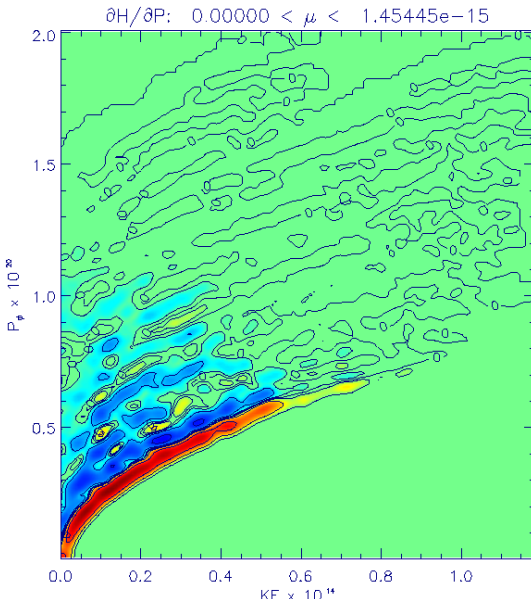
# High-Occupancy Bin Fits Are Good

$\lambda > 0$  Bin 1/25: 1,043,794 particles

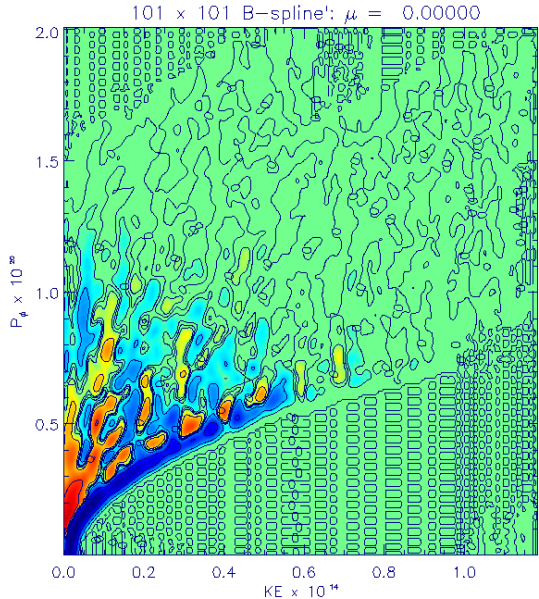
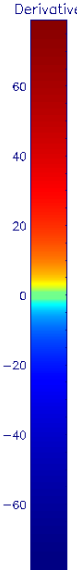
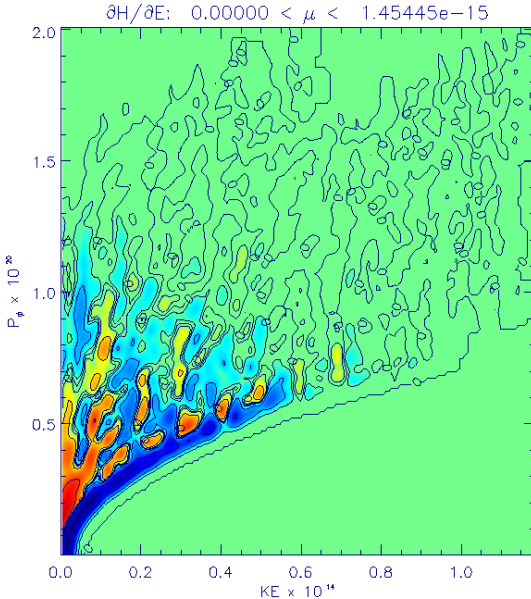


$$\sigma_P = \sigma_E = 1.334$$

# Gradients match as well



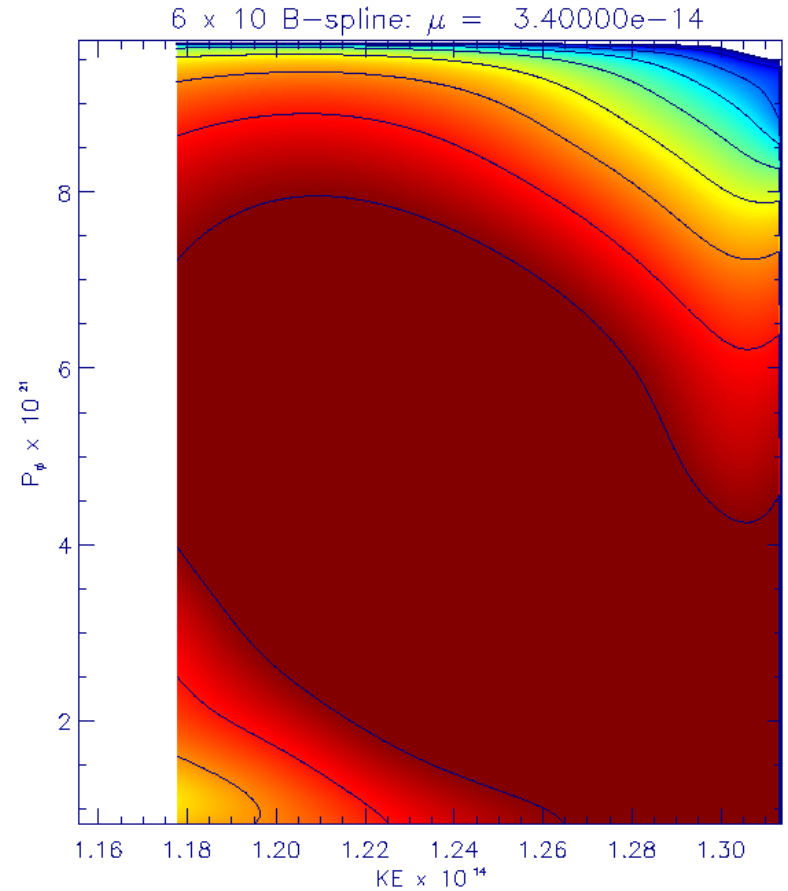
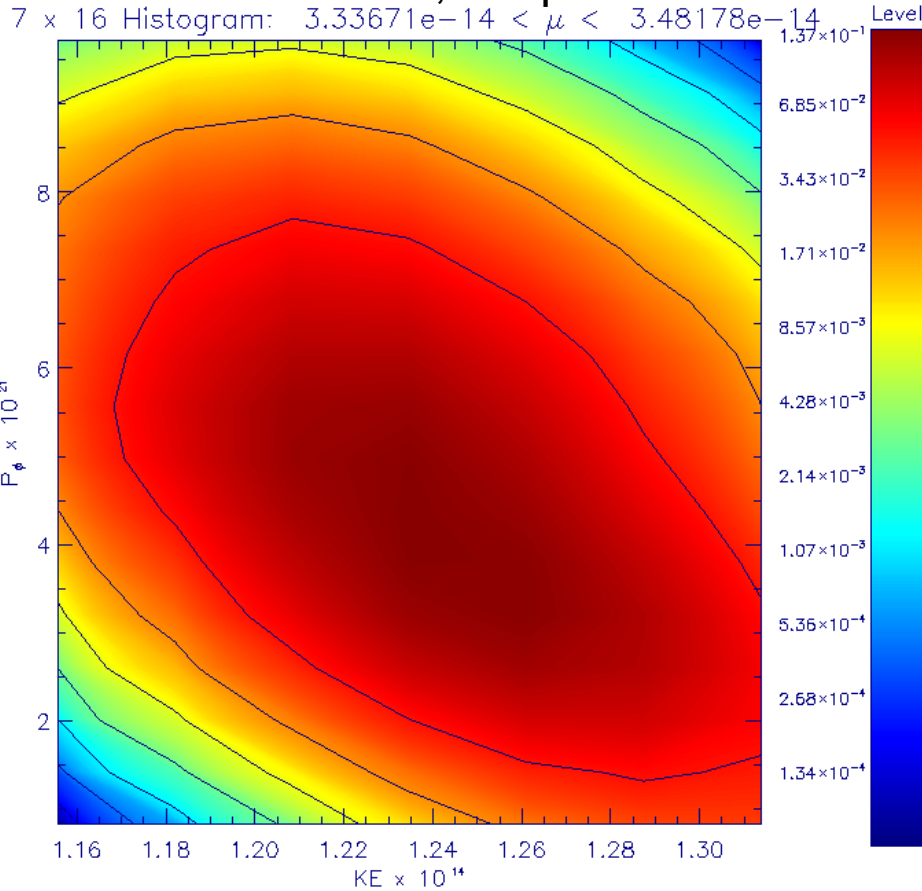
$$\frac{\partial f^+}{\partial P_\phi}$$



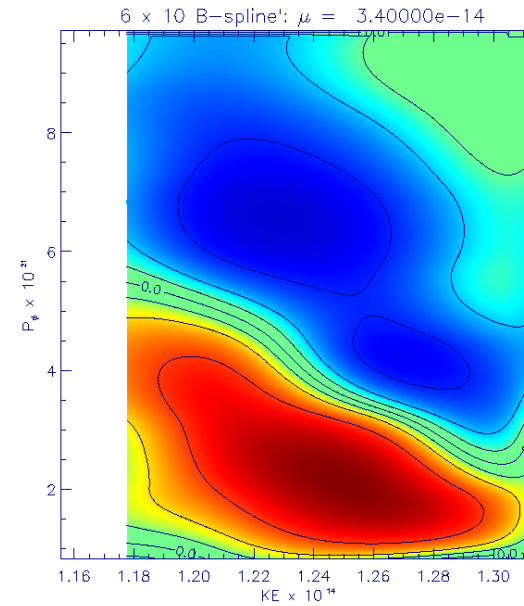
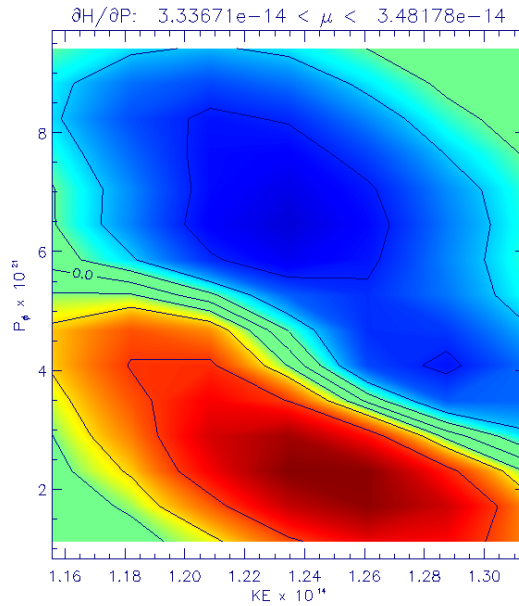
$$\frac{\partial f^+}{\partial E}$$

# Low-Occupancy Fits are Adequate

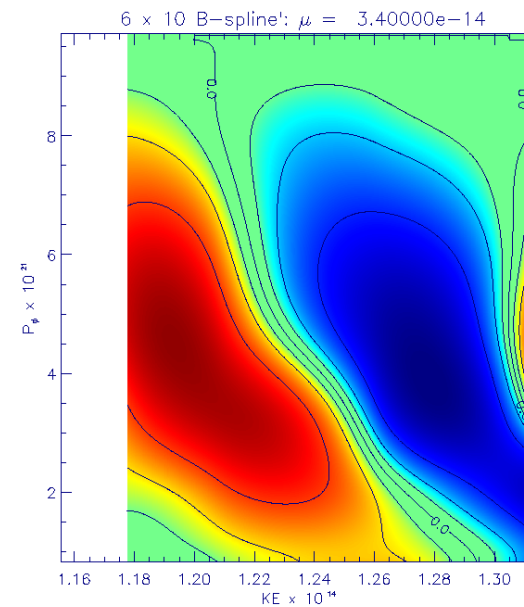
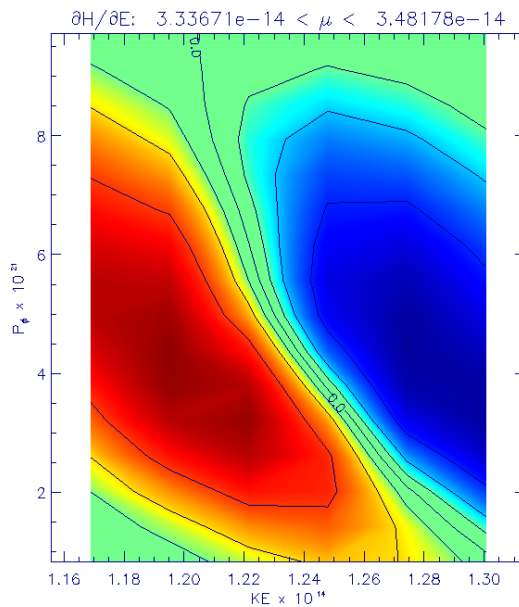
$\lambda < 0$  Bin 24/25: 1,208 particles



# Gradients are Smoothed



$$\frac{\partial f^-}{\partial P_\phi}$$



$$\frac{\partial f^-}{\partial E}$$



# Reconstruction Routines

## Fortran Interface:

pspline\_init\_()

*Reads particle spline coefficients from text file, initializes data structures to store them. Returns number of  $\mu$  bins for each orbit type.*

getpsplinebounds\_()

*Returns upper and lower bounds of particle distribution function space in each of the three constants of motion  $P_\varphi, \mu, E$ . (For  $E$ ,  $\min|E/\mu|=B_{min}$  sets the lower bound.)*

getpdf\_()

*Returns normalized  $f_t(P_\varphi, \mu, E)$ , where  $t$  is the orbit type (co, counter, or trapped).*

getpdfd\_()

*Returns normalized  $f_t(P_\varphi, \mu, E)$ ,  $\partial f_t/\partial P_\varphi$ , and  $\partial f_t/\partial E$ .*

pspline\_free\_()

*Frees up all storage associated with particle spline data structures.*

# Future Work

- Refine choices of particles/bin, smoothing kernel, uniformity of knots, splines/bin.
- Find better constraints for nonnegative spline values in fit.
- Improve accuracy of derivative around confined/lost boundary.
- Integrate sampling, coordinate conversion, Jacobian calculation, binning, and spline fitting components into a single utility.
- Test results in M3D-K
- Add flexibility for output to other SWIM codes.