

User manual for Chebsampling

Xin An^a, Anton Artemyev^a, Vassilis Angelopoulos^a, San Lu^b, Philip Pritchett^c, Viktor Decyk^c

^a*Department of Earth, Space and Planetary Sciences, University of California, Los Angeles, CA, 90095, USA*

^b*School of Earth and Space Sciences, University of Science and Technology of China, Hefei, China.*

^c*Department of Physics and Astronomy, University of California, Los Angeles, CA, 90095, USA*

Abstract

This document provides a manual for the code **Chebsampling**, a computationally efficient, robust tool based on inverse transform sampling with function approximation by Chebyshev polynomials. We provide details on the code structure and how to run the code to sample general distribution functions.

1 The main program (`main.f90`) is built on four modules, including `invsampling`,
2 `ppush2`, `distr`, and `pplib2`. Module `invsampling` contains major subrou-
3 tines implementing inverse transform sampling with function approximation
4 by Chebyshev polynomials. Module `ppush2` contains auxiliary subroutines
5 handling the domain partition and the deposition of particles. Module `distr`
6 contains subroutines generating original distribution data, which can be cus-
7 tomized by users for any desired distribution function. Module `pplib2` is
8 the basic parallel library for MPI communications inherited from the UPIC
9 framework [1]. Below we go through the workflow in the main program and
10 list the major subroutines in each module.

11 1. `main.f90`

12 We first list the key parameters and arrays in the main program. The
13 parameters can be modified by users for specific applications.

*Corresponding author.

E-mail address: xinan@epss.ucla.edu

- 14 • Parameter **nvp** defines the number of processors for massively parallel
15 computers. The special case of single processor **nvp** = 1 is handled.
16 In our implementation, this parameter must be less than the number
17 of grids in the *y* direction (**ncy**) and the number of particles in the *y*
18 direction (**np_y**).
 - 19 • Parameters **ncx** and **ncy** define the number of grids in the *x* and *y*
20 directions, respectively.
 - 21 • Parameters **np_x** and **np_y** define the number of particles in the *x* and *y*
22 directions, respectively.
 - 23 • Parameter **tol** controls the relative accuracy in chopping the Cheby-
24 shev coefficients.
 - 25 • Parameter **eps** controls the absolute accuracy in root finding with the
26 bisection method.
 - 27 • Array **fx_y** holds the original distribution data on grids.
 - 28 • Array **part** holds the particle data that are sampled from **fx_y**.
 - 29 • Array **density** holds the distribution function deposited from the sam-
30 pled particle data **part**. This array is for diagnostic purpose and should
31 be compared with **fx_y** to test the accuracy of our sampling method.
 - 32 • Array **edges** holds the lower and upper bounds of the domain partition.
- 33 A sequence of call statements gives the workflow of the main program.
34 Their functionalities are explained as follows.
- 35 • call **PPINIT2** initializes parallel processing for Fortran 90.
 - 36 • call **lembege_pellat** generates distribution data **fx_y** for the nonpo-
37 larized Lembege-Pellat current sheet. Note that users can define their
38 own distribution functions in Module **distr** and invoke these functions
39 here.
 - 40 • call **pfedges2** partitions the computational domain in the *y* direc-
41 tion so that each subdomain has approximately the same number of
42 particles.

- 43 • call `pp_inv_sampling_2D` generates particle data `part` from the dis-
44 tribution `fxy`.
- 45 • call `ppgpost2l` deposits density (on grid) from particle data `part`.
- 46 • call `PPNLAGUARD2L` adds data from guard cells in nonuniform parti-
47 tions for `density`.
- 48 • call `PPNLCGUARD2L` copies data to guard cells in nonuniform partitions
49 for `density`.
- 50 • call `PPWRVNDATA2` collects distributed nonuniform data `density` and
51 writes to a Fortran unformatted file.

52 2. Module `invsampling`

53 This module, central to `Chebsampling`, implements the discrete Cheby-
54 shev transform and uses this formalism in the root-finding problem of inverse
55 transform sampling. A 1D FFT library `mfft1` (inherited from the UPIC
56 framework) is used in the discrete Chebyshev transform.

- 57 • Subroutine `pp_inv_sampling_2D` performs inverse transform sampling
58 for general 2D distribution functions.
- 59 • Subroutine `inv_sampling_1D` performs inverse transform sampling for
60 general 1D distribution functions.
- 61 • Function `my_cumsum` calculates the cumulative distribution (CDF) from
62 the probability distribution function (PDF) based on grids.
- 63 • Function `inv_bisection` finds the root for $CDF(x) = y$ using the
64 bisection method, where x and y are vectors.
- 65 • Function `cheb_clenshaw` evaluates the Chebyshev sum $\sum_{k=1}^N c_k T_k(x)$
66 using the Clenshaw algorithm, where c_k and T_k ($k = 1, \dots, N$) are
67 Chebyshev coefficients and Chebyshev polynomials, respectively. Here
68 x is implemented as a vector.
- 69 • Subroutine `cheb_coeff` transforms function data to Chebyshev coeffi-
70 cients.

- 71 • Subroutine `chebpts` constructs Chebyshev points.
- 72 • Subroutine `map_unit_to_grid` maps any point on the unit interval
73 $[-1, 1]$ to the grid coordinate $[0, N_g]$.
- 74 • Subroutine `map_grid_to_unit` maps any point on the grid coordinate
75 $[0, N_g]$ to the unit interval $[-1, 1]$.
- 76 • Subroutine `interp1d` performs linear interpolation on 1D grid.
- 77 • Subroutine `DCT` performs discrete Chebyshev transform.
- 78 • Subroutine `cheb_standardChop` truncates Chebyshev coefficients to a
79 prescribed level of accuracy.

80 3. Module `ppush2`

- 81 • Subroutine `pdcomp2` partitions the computational domain in the y di-
82 rection in such a way that each subdomain contains the same number
83 of grids.
- 84 • Subroutine `pfedges2` partitions the computational domain in the y
85 direction in such a way that each subdomain contains the same number
86 of particles. Subroutines from Module `invsampling` are used here.
- 87 • Subroutine `ppgpost21` deposits density from particle data.

88 4. Module `distr`

89 Users can add desired distribution functions in this module. As described
90 in the main manuscript, this module currently contains density distributions
91 for nonpolarized and polarized Lembege-Pellat current sheets, as well as three
92 non-Maxwellian velocity distributions in the solar wind and the terrestrial
93 magnetosphere.

94 5. Module `pplib2`

95 `pplib2` is the parallel library for MPI communications in the UPIC frame-
96 work. Relevant documentations can be found in Ref. [1] and in the source
97 code.

⁹⁸ **References**

- ⁹⁹ [1] V. K. Decyk, UPIC: A framework for massively parallel particle-in-cell
¹⁰⁰ codes, Computer Physics Communications 177 (1-2) (2007) 95–97.