User manual for `Chebsampling`

Xin An[a], Anton Artemyev[a], Vassilis Angelopoulos[a], San Lu[a,1], Philip Pritchett[b], Viktor Decyk[b]

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

[b]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.

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

1. `main.f90`

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

*Corresponding author.
E-mail address: xinan@epss.ucla.edu
[1]Present address: University of Science and Technology of China, Hefei, China.

- Parameter `nvp` defines the number of processors for massively parallel computers. The special case of single processor `nvp` = 1 is handled. In our implementation, this parameter must be less than the number of grids in the y direction (`ncy`) and the number of particles in the y direction (`npy`).

- Parameters `ncx` and `ncy` define the number of grids in the x and y directions, respectively.

- Parameters `npx` and `npy` define the number of particles in the x and y directions, respectively.

- Parameter `tol` controls the relative accuracy in chopping the Chebyshev coefficients.

- Parameter `eps` controls the absolute accuracy in root finding with the bisection method.

- Array `fxy` holds the original distribution data on grids.

- Array `part` holds the particle data that are sampled from `fxy`.

- Array `density` holds the distribution function deposited from the sampled particle data `part`. This array is for diagnostic purpose and should be compared with `fxy` to test the accuracy of our sampling method.

- Array `edges` holds the lower and upper bounds of the domain partition.

A sequence of call statements gives the workflow of the main program. Their functionalities are explained as follows.

- `call PPINIT2` initializes parallel processing for Fortran 90.

- `call lembege_pellat` generates distribution data `fxy` for the nonpolarized Lembege-Pellat current sheet. Note that users can define their own distribution functions in Module `distr` and invoke these functions here.

- `call pfedges2` partitions the computational domain in the y direction so that each subdomain has approximately the same number of particles.

- `call pp_inv_sampling_2D` generates particle data `part` from the distribution `fxy`.

- `call ppgpost2l` deposits density (on grid) from particle data `part`.

- `call PPNLAGUARD2L` adds data from guard cells in nonuniform partitions for `density`.

- `call PPNLCGUARD2L` copies data to guard cells in nonuniform partitions for `density`.

- `call PPWRVNDATA2` collects distributed nonuniform data `density` and writes to a Fortran unformatted file.

2. Module `invsampling`

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

- Subroutine `pp_inv_sampling_2D` performs inverse transform sampling for general 2D distribution functions.

- Subroutine `inv_sampling_1D` performs inverse transform sampling for general 1D distribution functions.

- Function `my_cumsum` calculates the cumulative distribution (CDF) from the probability distribution function (PDF) based on grids.

- Function `inv_bisection` finds the root for $CDF(x) = y$ using the bisection method, where x and y are vectors.

- Function `cheb_clenshaw` evaluates the Chebyshev sum $\sum_{k=1}^{N} c_k T_k(x)$ using the Clenshaw algorithm, where c_k and T_k $(k = 1, \cdots, N)$ are Chebyshev coefficients and Chebyshev polynomials, respectively. Here x is implemented as a vector.

- Subroutine `cheb_coeff` transforms function data to Chebyshev coefficients.

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

3. Module `ppush2`

- Subroutine `pdcomp2` partitions the computational domain in the y direction in such a way that each subdomain contains the same number of grids.
- Subroutine `pfedges2` partitions the computational domain in the y direction in such a way that each subdomain contains the same number of particles. Subroutines from Module `invsampling` are used here.
- Subroutine `ppgpost2l` deposits density from particle data.

4. Module `distr`

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

5. Module `pplib2`

`pplib2` is the parallel library for MPI communications in the UPIC framework. Relevant documentations can be found in Ref. [1] and in the source code.

6. How to run the code

We provide a makefile and an example PBS script for compiling and running the program. To sample a new customized distribution function, users should add it to Module `distr` and modify the corresponding call statement in the main program.

References

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