Universitat Politècnica de Catalunya
Department of Civil and Environmental Engineering
Hydrogeology Group

GPKDE: Fortran code for Grid Projected Kernel Density Estimation of Discrete Particle Distributions

Documentation of Input-Output

Version 1.0.0

Rodrigo Pérez-Illanes

Daniel Fernàndez-Garcia

Barcelona
2023

Contents

Chapter 1

Introduction

This report contains information about the configuration of input files and output structures for the program GPKDE. The software performs Grid Projected Kernel Density Estimation of a discrete distribution of points in one, two or three dimensions, based on the methodology presented in Sole-Mari *et al.* (2019). This code is a new implementation of the method employing the Fortran programming language, parallelized with the OpenMP library and with compatibility for weighted particle distributions.

The code is delivered with a user interface (`GPKDE.F90`) that enables its use as an independent program, interpreting a configuration file defining the loading of data coordinates, the kernel bandwidth optimization and other relevant program parameters. This document provides the configuration instructions for this specific interface and users pursuing the integration into an external program are encouraged to explore the interface file. The main reconstruction functionalities are grouped into modules and can be made available to external software by employing the files:

- `GridProjectedKDE.F90`: provides the reconstruction methodology, optimization for bandwidth selection, interfaces for computing density of a given dataset and writing output files.
- `Histogram.F90`: Computation of histograms with uniform and non-uniform weights.
- `KernelMultiGaussian.f90`: Kernel functions employed in the program.
- `GridCell.f90`: Utils for storing specific parameters for a grid cell while computing kernel convolutions.
- `Precision.F90`: Module configuring the floating point precision across the program.
- `Constants.f90`: Constant values employed throughout the program, following the selected floating point precision.

Source code repository is available via Github[1] and example applications are provided illustrating different use cases of the program. The channel is the main source of information regarding bug reports and program updates.

[1] https://github.com/upc-ghs/gpkde

Chapter 2

Input

The program requires a single simulation file, in which information necessary for the configuration of the reconstruction process is provided. Parameters defining the reconstruction grid, controls over the kernel bandwidth optimization and other specifications are indicated in this file. The command line interface provides some arguments and utilities specified later in this chapter.

2.1 Simulation file

Main entry point for a GPKDE simulation. A program run is executed following the simplified command line instruction

```
gpkde simfile
```

The structure of the simulation file and illustrative values are shown in Figure (2.1). Some of the input values are optional, and eventually, the interpretation of others is determined by values given to preliminary parameters. The specific interpretation of parameters and possible values for these variables are shown in Tables (2.1) and (2.2).

The simulation file begins by indicating the name of the file where the particles/points data is stored, and several alternatives for the interpretation of the data structure are provided (`InputDataFormat`). The default format consists of a text-plain document containing the three-dimensional particle coordinates `(x y z)`. Users can specify that the input data also includes an specific particle weight `(x y z w)`. The user can also choose to indicate the number of columns in this file, that is, an input file could be interpreted, for example, as `(x y)` or `(x y w)`. In the last case, although the file contains three columns, only the first two columns are interpreted as coordinates and the last would be interpreted as weight. Regardless of the input data structure, the reconstruction module is implemented expecting an array of particles with three-dimensional coordinates, so in all cases the data will be organized following this principle. All of the combinations of data structures can be interpreted from text-plain or binary files. In this last case, users need to be aware of the floating-point precision employed while writing the data, and be consistent with the floating-point precision employed while compiling the program, otherwise the loading of the data file might be inconsistent. Following the specification of the input data structure, the user can indicate the number of lines in the data file, which if given, will lead to a faster loading of the data points. If not, the program will infer how many particles are provided by first counting the number of data points in the input file.

```
0 : Optional comment line        | # GPKDE configuration file |
1 : DataFileName                 | particles.csv              |
2 : i.   InputDataFormat         | 0   3   10000   1.0    0   |
    ii.  NDimCols                | density.out     0      0   | 3: i. OutFileName ii. ColumnFormat iii. FileFormat
    iii. NPoints                 | 0.0    0.0   0.0   0       | 4: i. DomainOrigin [x y z] ii. SlicedDimension
    iv.  UniformWeight           | 100.0  50.0  10.0  1  0.05 | 5: i. DomainSize [x y z] ii. GridAlloc iii. BorderFrac
    v.   EffectiveWeightFormat   | 1.0    1.0   1.0   0  0.9  | 6: i. BinSize [x y z] ii. AutomaticBin iii. BinFactor
7 : i.   NOptLoops               | 10     0                   |
    ii.  ExportOptVars           | 0      1E-03               | 8: i.   SkipErrorConvergence
9 : i.   KernelDatabase          | 1      0      0            |    ii.  RelativeConvergence
    ii.  BoundKernelFormat       | 1.0    0.1    10.0         |10: KDB (MinHD DeltaHD MaxHD)
    iii. IsotropicKernels        | 1      5.0                 |11: i.   InitialSmoothingFormat
12: InitialSmoothingArray [x y z]| 5.0    1.0    1.0          |    ii.  SizeFactor
13: AdvancedOptions              | 1                          |
14: i.   MinRoughnessFormat      | 1      1E-04   1.0         |
    ii.  MinRefRoughness         | 0.9                        |15: IsotropicThreshold
    iii. MinRoughnessLScale      | 0                          |16: UseGlobalSmoothing
```

Figure 2.1: Illustrative GPKDE simulation file. Example parameters are enclosed between vertical dividers and their names are given in left/right columns. Detailed specifications are given in Tables (2.1) and (2.2).

In case that the data structure considers only coordinates, the user can specify a parameter that scales the particle density, representing a uniform particle weight (`UniformWeight`). In practice, this parameter is useful to transform the particle density to other interpretations (for example mass concentration). In case that an specific (eventually non-uniform) particle weight is included, the user can select one of several alternatives determinig the protocol to transform the weighted-histogram into an equivalent count of particles (`EffectiveWeightFormat`). The default format follows the method for weighted distributions of Kish (1965, 1992), where an effective number of particles is computed, and in combination with the total mass, allows to obtain a unique effective particle weight. This last quantity is employed to transform the mass histogram into an equivalent particle count, which is employed to calculate particle densities during the kernel bandwidth selection. A second (similar) format transforms the weighted-histogram into an equivalent count of particles by means of the simple average particle weight. These two alternatives perform a final scaling operation over the estimated particle density in order to transform back to mass density. The third format performs the bandwidth selection by means of the real particle count and a final reconstruction is performed over the weighted-histogram, given a distribution of kernel sizes. The last format computes an effective number of particles for each grid-cell (Kish, 1965, 1992), which is employed for the purposes of bandwidth selection, and similar to previous case, a final reconstruction over the weighted histogram is performed. This alternative has been observed to be the more suitable for the reconstruction of datasets with low number of particles and high variability of the individual weights.

After the input data specification, the simulation file continues with the information configuring the output file, where density will be written. Besides the file name, some options are given to users in order to control the structure of the output (as detailed in section 3.1), and the file format (text-plain or binary).

Table 2.1: Simulation parameters for the GPKDE configuration file.

Id	Parameter	Type	Values
1	DataFileName	string	The name of the input data file (max. 200 characters).
2.i	InputDataFormat	int	0: Data file read as (x,y,z). 1: Data file with weights (x,y,z,w). 2: Binary data file (x,y,z). 3: Binary data file with weights (x,y,z,w). 4: Data file read as (1:NDimCols). 5: Data file with weights (1:NDimCols,w). 6: Binary data file (1:NDimCols). 7: Binary data file with weights (1:NDimCols,w).
2.ii	NDimCols	int	Read only if InputDataFormat>=4. The number of input columns representing coordinates/dimensions.
2.iii	NPoints	int	The number of points. If NPoints=0, is infered from file.
2.iii	UniformWeight	float	Uniform weight if InputDataFormat=0,2,4,6.
2.iv	EffectiveWeightFormat	int	Read only if InputDataFormat=1,3,5,7. 0: Effective sample size and a domain effective weight. 1: Equivalent histogram by means of the average weight. 2: Bandwidth selection from particle positions, real histogram. 3: Effective weight for each cell.
3.i	OutFileName	string	The output file where density is written (max 200 characters).
3.ii	ColumnFormat	int	0: Bin id and densities (ix,iy,iz,rho,hist). 1: Bin id, coordinates and densities (ix,iy,iz,x,y,z,rho,hist). 2: Coordinates and densities (x,y,z,rho,hist).
3.iii	FileFormat	int	0: Output file as text-plain. 1: Output file as binary.
4.i	DomainOrigin	float	Coordinate x y z for the origin (west, south, bottom).
4.ii	SlicedDimension	int	0: Disable sliced reconstruction, default. 1, 2 or 3: Reconstruction slicing the given dimension.
5.i	DomainSize	float	Dimensions/size x y z of the domain.
5.ii	GridAlloc	int	0: Grids allocated according to domain size. 1: Grids allocated according to min/max coordinates.
5.iii	BorderFrac	float	Read if GridAlloc=1. Defines an extra distance for the size of the reconstruction grid, as a fraction of the extent on each dimension.
6.i	BinSize	float	Dimensions/size x y z of the grid-cells.
6.ii	AutomaticBin	int	0: use the given bin size, default. 1: select bin size based on multidimensional Scott (2015) rule. 2: select bin size with Freedman & Diaconis (1981) rule. Only 1D.
6.iii	BinFactor	float	Factor $\in [0, 2]$ multiplying the automatic bin size.
7.i	NOptLoops	int	The maximum number of optimization loops.
7.ii	ExportOptVars	int	0: Do not export optimization variables. 1: Write a file per optimization loop with variables.

Following, the program continues with the interpretation of the reconstruction domain. The reconstruction module will compute the index of the bin/cell to which a point belongs, based on the grid specifications. The domain origin, domain size and the bin sizes are required for creating a grid representation, which ultimately determines the particles/points that will be considered during the reconstruction process, taking into account only those within the boundaries delimited by the domain specification. The domain origin should be considered as the most (west, south, bottom) point in the system. From here, the domain size extends toward the (north, east, top), respectively. The domain specifications can be single-handedly specified by the user, but the program also provides some automatic alternatives assisting the definition of the reconstruction grid. For example, two simple data-based rules for selecting the bin size are implemented: the Scott's rule for the optimal bin of a normal distribution (Scott, 1979, 2015) which is compatible for multidimensional problems, and the Freedman & Diaconis (1981) rule which is available only for 1-dimensional problems. An optional parameter allow users to specify whether the grids employed during reconstruction should be allocated following the given domain dimensions or adapted to the coordinates of the particles/-points distribution (`GridAlloc`). Allocating according to the particle coordinates is usually more efficient in terms of memory, as the grid size is determined based on the bounding-box coordinates of the distribution, plus an additional border distance defined as a fraction of the extent on a given dimension. This approach avoids the sometimes unnecessary allocation of very large grids containing a large number of cells without information. While allocating with this approach, the additional border is necessary because kernels will be distributing information to cells beyond the histogram limits. An optional program parameter allows the user to perform reconstruction in a sliced manner (`SlicedDimension`). This means that the reconstruction of a d-dimensional domain is performed with $(d-1)$-dimensional kernels, performed independently for each slice of the reconstruction grid. This approach can be useful while considering the reconstruction in highly anisotropic domains, where the extent in one of the dimensions is markedly smaller with respect to the others.

The program continues with the interpretation of parameters controlling the bandwidth optimization loop. A maximum number of loops is required, and a flag is provided to indicate whether the intermediate program variables of a given loop should be exported. This functionality can be useful for debugging purposes or simply to monitor the evolution of the quantities employed for determining the optimal kernel sizes. The export process writes loop specific files, with names that concatenate the output file name and the loop number (more details in section 3.3). Also as a control of the optimization process, the user can specify whether the error convergence verification should be performed. In case that this check is skipped, then the program will calculate until reaching the maximum number of optimization loops. If the error checks are preserved, then the user can specify a value for the relative error convergence parameter, which defines the threshold for deciding whether convergence has been achieved or not.

Following, the simulation file requires information for the configuration of kernels. The user can indicate whether kernels should be computed in real-time or precalculated and stored on a kernels database. For multidimensional reconstruction, it is generally recommended to preallocate a database of kernels. It has been observed that this functionality provides improvements in computational times with respect to real-time calculations, sacrificing some accuracy on the kernel values, which in practice could be controlled by the level of discretization of the database. The sizes of pre-

allocated kernels are specified in terms of non-dimensional bandwidths, defined as the ratio between the kernel bandwidth and the bin size. While using a kernel database, the user needs to provide the minimum non-dimensional smoothing, a step, and the maximum value (`MinHD DeltaHD MaxHD`). For one and two dimensional problems, it has been observed that the program is able to provide fast reconstruction even while employin the real-time calculation of kernels.

An optional parameter allows users to specify a protocol for limiting the size of kernels. The first alternative, bound kernel sizes based on domain restrictions and is the default format. In the upper bound, the kernel bandwidth cannot grow larger than a fraction $(0.75-1)$ of the domain size, which is a good approach for preventing inconsistencies while correcting kernels by boundary reflection,

Table 2.2: Simulation parameters for the GPKDE configuration file (continued).

Id	Parameter	Type	Values
`8.i`	`SkipErrorConvergence`	`int`	`0`: Verifies convergence criteria and break. `1`: Skip convergence verification.
`8.ii`	`RelativeConvergence`	`float`	Threshold of relative change between subsequent loops.
`9.i`	`KernelDatabase`	`int`	`0`: Kernels are computed in real time. `1`: Kernels are precalculated and stored in database.
`9.ii`	`BoundKernelFormat`	`int`	`0`: Bound kernels size based on domain restrictions. `1`: Bound kernels size based on `MinHD` and `MaxHD`. `2`: Unbounded kernel sizes.
`9.iii`	`IsotropicKernels`	`int`	`0`: Kernels are anisotropic. `1`: Kernels are isotropic.
`10`	`KDB`	`float`	Range of non-dimensional smoothing for the database. `MinHD,MaxHD` read if `KernelDatabase=1` or `BoundKernelFormat=1`. `DeltaHD` only read if `KernelDatabase=1`.
`11.i`	`InitialSmoothingFormat`	`int`	`0`: First bandwidth obtained from Silverman expression. `1`: First bandwidth as a factor multiplying bin size. `2`: First bandwidth given by user as `x y z` array.
`11.ii`	`SizeFactor`	`float`	If `InitialSmoothingFormat=0`: if given a `SizeFactor!=0`, then multiply the automatically estimated first bandwidth. If `InitialSmoothingFormat=1`, multiply the characteristic bin size.
`12`	`InitialSmoothingArray`	`float`	Initial bandwidth if `InitialSmoothingFormat=2`.
`13`	`AdvancedOptions`	`int`	If `1`, enables the interpretation of advanced parameters.
`14.i`	`MinRoughnessFormat`	`int`	`0`: Minimum roughness assuming Gaussian distribution. `1`: Limit determined using `14.ii` and `14.iii`. `2`: Uses `14.ii` as minimum value. `3`: Minimum roughness is zero.
`15`	`IsotropicThreshold`	`float`	If the fraction between a directional roughness and isotropic net roughness is above this threshold, then then net roughness is corrected by isotropic estimate.
`16`	`UseGlobalSmoothing`	`int`	If `1`, optimal smoothing as global isotropic.

avoiding the occurrence of multiple reflections, which at this stage are not explicitly implemented. The lower bound of the kernel size is established in terms of a minimum number of cells, forcing the kernel bandwidth to have at least two cells at each side of its center, avoiding extremelly small kernels. In the second bounding format, the program makes use of the values for `MinHD` and `MaxHD`. In case that these values were not given because the program is not employing a kernel database, then both will be explicitly read. In the third case, kernel sizes are unbounded. An optional flag is provided to indicate whether in multidimensional problems kernels shall be considered as isotropic or anisotropic, being the latter the default. The process of kernel bandwidth selection begins from a given initial value, which is employed to initialize densities, and users have different alternatives to indicate the initial bandwidth (`InitialSmoothingFormat`). This can be estimated automatically from the expression for a Gaussian distribution (Silverman, 1986), or as a factor multiplying the cell sizes, or simply by providing an array with the bandwidths for each direction.

Until this point, information about the previously described parameters is expected to be provided in the input simulation file. An additional set of options are available, controlling some advanced aspects of the reconstruction process. If the interpretation of the advanced options is enabled, the input file will be read until the last given parameter and the remaining unespecified values will remain as default. The first advanced control, allows to define bounding values for the net roughness, which can act as an additional alternative to control kernel sizes. A minimum roughness can be defined from assuming a Gaussian distribution and a predefined value for the non-dimensional roughness is employed, which was determined from the analytical study of the Gaussian problem. To obtain this estimation, the program internally employs the standard deviation of the distribution and the maximum density, combined with the default non-dimensional roughness, to define a lower bound. In the second case, the user provides a minimum relative roughness and a characteristic length scale. The program will use these values and the maximum density to obtain the lower bound. In the third case, the program reads a given value of roughness which is used as the lower bound. A last alternative defines that the roughness is not explicitly unbounded, although in any case the program will compute the kernel bandwidth only for non-zero net roughness.

The next advanced specification allows the user to indicate a threshold value defining the relative fraction of a directional roughness with respect to the sum of the main roughnesses, above which the program applies a correction to the estimated net roughness in case of using anisotropic kernels. This is useful for cases where the particle distribution presents a strong symmetry in one direction, possibly leading to very small values of the net roughness while calculating with the anisotropic expression. In essence, if any of the directional roughnesses explains more than the `IsotropicThreshold` of the sum of the main directional roughnesses, the estimated net roughness is corrected and recalculated with the isotropic expression.

The last parameter is a flag to indicate whether the kernel bandwidth should follow a global optimal smoothing for the reconstruction. This means that the net roughness is not a local integral, instead is replaced by a domain integral, and the optimal smoothing is isotropic and homogeneous for the whole domain, obtained with a global expression. This option overrides the locality principles upon which most of the program is built and it can be useful in rare cases of extremelly sharp gradients or close-to-uniform distributions.

2.2 Command line interface

Some basic operations can be managed from the command line interface implemented for the main program. The basic instructions for using the command line can be requested with the command `gpkde --help` or simply `gpkde -h`, which will show in console the following message:

```
GPKDE version *.*.*
Program compiled Apr 12 2023 19:44:24 with GFORTRAN compiler (ver. *.*.*)

Fortran code for Grid Projected Kernel Density Estimation of discrete particle distributions

usage:

  gpkde [options] simfile

options:

  -h         --help                Show this message
  -l  <str>  --logname    <str>    Write program logs to <str>
  -nl        --nolog               Do not write log file
  -np <int>  --nprocs     <int>    Run with <int> processes
  -p         --parallel            Run in parallel
  -v         --version             Show program version

For bug reports and updates, follow:
  https://github.com/upc-ghs/gpkde
```

Figure 2.2: Help message from the GPKDE program.

Chapter 3

Output

The current version of the program can generate up to three different output files, which are detailed in the following.

3.1 Density

This is the main output from the program. It contains the reconstructed density and associated grid indexes. The file contains the information only for those cells with a non-zero density. This implies that cells without particles, but with an estimated density, will also be written in this file. The histogram density is also included in the last column.

The specific structure of this file can be controlled with the input parameter `3.ii ColumnFormat` and besides the cell indexes, the output can be requested including the center coordinates of the associated bin. The parameter `3.iii FileFormat` determines if the file is written in text-plain or binary format. In this regard, users opting for writing the output files in binary, should keep in mind the floating-point precision employed during the program compilation (`real32 or real64`), because binary variables written to the file will preserve this precision. This applies only to variables of type `float`.

The column combinations can adopt the following forms:

`ColumnFormat=0`: Bin id's, smoothed and histogram densities.

```
1 : idBinX    (int)
2 : idBinY    (int)
3 : idBinZ    (int)
4 : Density   (float)
5 : Histogram (float)
```

`ColumnFormat=1`: Bin id's, center cell coordinates and densities.

```
1 : idBinX     (int)
2 : idBinY     (int)
3 : idBinZ     (int)
4 : X          (float)
5 : Y          (float)
6 : Z          (float)
7 : Density    (float)
8 : Histogram  (float)
```

`ColumnFormat=2`: Center cell coordinates and densities.

```
1 : X         (float)
2 : Y         (float)
3 : Z         (float)
4 : Density   (float)
5 : Histogram (float)
```

3.2 Log file

This file is written during the interpretation of parameters and reconstruction process, containing a summary of the most relevant information related to the program execution. Contains a report of the program workflow based on the user input parameters. Also contains a summary of the relevant metrics of the optimization process. This file is in general a good source of information for post-processing purposes.

By default, every time the program is executed, this file is written as `gpkde.log`. As shown in section (2.2), the user can make use of the command line arguments and indicate whether to skip the generation of this file (`--nolog, -nl`), or change the name of the log file (`--logname, -l`).

3.3 Optimization variables

File generated when users specify that optimization variables should be exported. One text-plain file is generated for each optimization loop following the naming convention `OutFileName`+`loopId`. In contrast to the density file, optimization variables are written only for those cells that contain particles. The structure of this file is as follows:

```
1 : BinX             (int)     8 : ShapeFactorX          (float)     15: CurvatureBandwidthZ  (float)
2 : BinY             (int)     9 : ShapeFactorY          (float)     16: AveragedDensity      (float)
3 : BinZ             (int)     10: ShapeFactorZ          (float)     17: RoughnessXX          (float)
4 : Density          (float)   11: KernelBandwidthScale  (float)     18: RoughnessYY          (float)
5 : KernelBandwidthX (float)   12: KernelSupportScale    (float)     19: RoughnessZZ          (float)
6 : KernelBandwidthY (float)   13: CurvatureBandwidthX   (float)     20: NetRoughness         (float)
7 : KernelBandwidthZ (float)   14: CurvatureBandwidthY   (float)
```

Bibliography

Freedman, D., & Diaconis, P. 1981. On the histogram as a density estimator:L 2 theory. *Zeitschrift fur Wahrscheinlichkeitstheorie und Verwandte Gebiete*, **57**(4), 453–476.

Kish, L. 1965. *Survey sampling*. John Wiley & Sons.

Kish, L. 1992. Weighting for unequal P_i. *Journal of Official Statistics*, **8**(2), 183–200.

Scott, D. W. 1979. On optimal and data-based histograms. *Biometrika*, **66**(3), 605–610.

Scott, David W. 2015. *Multivariate density estimation*. 2 edn. Wiley Series in Probability and Statistics. Chichester, England: Wiley-Blackwell.

Silverman, B. W. 1986. *Density estimation for statistics and data analysis*. Vol. 26. CRC press.

Sole-Mari, G., Bolster, D., Fernàndez-Garcia, D., & Sanchez-Vila, X. 2019. Particle density estimation with grid-projected and boundary-corrected adaptive kernels. *Advances in Water Resources*, **131**(Sept.), 103382.