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 main 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 bandwidth optimization process and kernel specifications are indicated in this file. The command line interface provides program arguments specified later in this chapter.

2.1 Simulation file

Main entry point for a GPKDE simulation. The program 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 controlled by values specified by the user. Specific meaning and possible options for these variables are shown in Tables (2.1) and (2.2).

The simulation file begins with the file name of the input data. The user needs to specify an input format which might be written only as coordinates `(x y z)` or also including an specific weight `(x y z w)`, in both text-plain or binary input formats. For the latter, users need to be aware of the floating point precision employed while writting the binary data file, 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 format specification, the user can indicate the number of lines in the data file, which if given will lead to 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. In the case the format considers only coordinates, the user can still specify a scaling parameter representing a uniform weight, which in practice can be useful to transform the particle density to other interpretations (for example mass concentration). In the case an specific particle weight is included, the user can specify how to interpret these values while transforming the weights histogram into an equivalent particle histogram. The default format follows the method for weighted distributions of Kish (1965, 1992), where an effective number of particles is computed at the domain level, from where an effective particle weight is obtained. The second format considers the bandwidth selection by transforming

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 bandwidth selection with the real particle histogram and a final reconstruction is performed over the weighted histogram once a bandwidth distribution was determined. The last format computes an effective number of particles for each cell (as in 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. The latter has been observed to be the more suitable method for the reconstruction of datasets with low number of particles and high weights variability.

After the input data specification, the simulation file continues with information about the output file where density will be written. Besides the file name, some options are given to select the information written to output files and file format, which could be text-plain or binary, as detailed in Section (3.1).

Following, the program continues with interpretation of the reconstruction domain. The program will compute the bin or cell to which a point belongs based on the relevant dimensions specified by the user. In this regard, the domain origin, domain size and the bin sizes are required for creating a grid representation. An optional parameter allow users to specify if the reconstruction and other grids for internal use shall be allocated following the domain dimensions or adapted to the given points distribution. The latter is usually more efficient in terms of memory as the grid size is determined based on the minimum and maximum coordinates of the distribution, plus an additional border distance defined as a fraction of the extent on a given dimension. This additional border is necessary because kernels are most likely distributing information to cells outside the histogram limits. The points to be considered in the reconstruction process are those within the boundaries delimited by the domain specification.

```
0 : Optional comment line        | # GPKDE configuration file |
1 : DataFileName                 | particles.csv              |
2 : i.   InputDataFormat         | 0    10000    1.0    0     |
    ii.  NPoints                 | density.out    0     0     | 3: i. OutFileName ii. ColumnFormat iii. FileFormat
    iii. UniformWeight           | 0.0    0.0   0.0           | 4: DomainOrigin [x y z]
    iv.  EffectiveWeightFormat   | 100.0  50.0  10.0  1  0.05 | 5: i.   DomainSize [x y z]
6 : BinSize [x y z]              | 1.0    1.0   1.0           |    ii.  GridAllocationFormat
7 : i.   NOptLoops               | 10     0                   |    iii. BorderFraction
    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 (MinHL DeltaHL MaxHL)
    iii. IsotropicKernels        | 1      5.0                 |11: i.   InitialSmoothingFormat
12: InitialSmoothingArray [x y z]| 5.0    1.0    1.0          |    ii.  BinSizeFactor
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.

Table 2.1: Simulation parameters for the GPKDE configuration file.

Id	Parameter	Type	Values
1	DataFileName	string	The name of the input data file.
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 with weights (x,y,z). 3: Binary data file with weights (x,y,z,w).
2.ii	NPoints	int	The number of points. If NPoints=0 infers from file.
2.iii	UniformWeight	float	Uniform weight if InputDataFormat=0,2.
2.iv	EffectiveWeightFormat	int	Read only if InputDataFormat=1,3. 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	OutputFileName	string	The output file where density is written.
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	InputDataFormat	int	0: Output file as text-plain. 1: Output file as binary.
4	DomainOrigin	float	Coordinate x y z for the origin.
5.i	DomainSize	float	Dimensions x y z of the domain.
5.ii	GridAllocationFormat	int	0: Grids allocated according to domain size. 1: Grids allocated according to min/max coordinates.
5.iii	BorderFraction	float	Read if GridAllocationFormat=1. Defines the extra distance for defining the size of reconstruction grid as a fraction of the extent on each dimension.
6	BinSize	float	Dimensions x y z of the cells.
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.
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 based on domain restrictions. 1: Bound kernels based on MinHL and MaxHL. 2: Unbounded kernels.
9.iii	IsotropicKernels	int	0: Kernels are anisotropic. 1: Kernels are isotropic.
10	KDB	float	Range of non-dimensional smoothing for the database. MinHL,MaxHL read if KernelDatabase=1 or BoundKernelFormat=1. DeltaHL only read if KernelDatabase=1.

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

Id	Parameter	Type	Values
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	BinSizeFactor	float	For initial bandwidth if InitialSmoothingFormat=1.
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.

After the grid specification, the program continues with the interpretation of parameters controlling the bandwidth optimization loop. More specifically, the maximum number of loops is required and a flag is provided to indicate whether the variables of a given loop should be exported or not. This can be useful for debugging purposes and essentially the quantities employed for determining the optimal kernel sizes are written to loop specific files, with names that concatenate the output file name and the loop number. Also as a control of the optimization process, the user can specify whether the error convergence verification should be performed. In case the latter is skipped, then the program will calculate until 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 kernels configuration. 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 the kernel database as it provides improvements in computational times with respect to real time calculations, sacrificing some accuracy on kernel values, which in practice can be also controlled by the level of discretization of the database. The latter is specified in terms of the non-dimensional smoothing, that is, the ratio between the kernel bandwidth and the bin size. In case of using the database, the user needs to provide the minimum non-dimensional smoothing, a step, and the maximum value (MinHL DeltaHL MaxHL). For one and two dimensional problems, it has been observed that the program is able to provide fast reconstruction while using real time calculation of the kernels.

An optional parameter allows users to specify a protocol for limiting the kernel sizes, with three different alternatives. The first, bound kernel sizes based on domain restrictions and is the default format. In terms of the upper bound, the kernel support cannot grow larger than half the domain

size, which is a good approach to prevent inconsistencies while correcting the kernel values by boundary reflection, avoiding potentially multiple reflections which are not explicitly implemented. The lower bound is established in terms of a minimum number of cells, by forcing kernels 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 `MinHL` and `MaxHL`, and in case these were not given because the program is not employing a kernel database, then both will be explicitly read. In the third case, kernels 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.

Input file continues with the specification for selecting the initial kernel bandwidth. In this regard, the user can specify whether an initial bandwidth should be estimated from the expression for Gaussian distributions of Silverman (1986), or as a factor multiplying the cell sizes, or by providing an array with the bandwidths for each direction.

Until this point, the previous parameters are expected to be provided, but an additional set of options is also given to control 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 preserve the default.

Following, options for bounding the net roughness are provided, which provides some additional alternatives for controlling kernel sizes. In the first case, a minimum roughness is established assuming a Gaussian distribution and a predefined value for the non-dimensional roughness is employed, determined from the theoretical analysis of the Gaussian problem. 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 a lower bound. In the third case, the program reads a given value of roughness which is used as the lower bound. In the last case, the roughness is not explicitly bounded, although the program will compute the 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, while using anisotropic kernels. This is useful for cases where the particle distribution presents a strong symmetry in one direction, possible leading to very small values of the net roughness, but with a clearly dominant direction. In essence, if any of the directional roughnesses explains more than the `IsotropicThreshold` of the sum of the main directional roughnesses, the net roughness is corrected.

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 (see Sole-Mari & Fernàndez-Garcia, 2018). 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 may 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 information only for those cells with a non-zero density. Notice that the latter implies that cells without particles, but with an estimated density, will also be written in this file. The histogram density is by default also included in the last column for comparison purposes.

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 cell 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 format, should take into account the floating point precision employed during program compilation (`real32 or real64`). While reading the output binary files from other programs, it should be enforced that the floating precision of the variables receving the data be the same than the floating precision with which the data was written, otherwise values will be inconsistent. 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 and parameters. It reports the program workflow based on user input parameters and 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 the file `gpkde.log` is written. Users can indicate that this file should not be generated or change its name by means of the command line parameters, as indicated in Section (2.2).

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 `OutputFileName+loopId`. In contrast to the density file, optimization variables are only written 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

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

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

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

Sole-Mari, G., & Fernàndez-Garcia, D. 2018. Lagrangian modeling of reactive transport in heterogeneous porous media with an automatic locally adaptive particle support volume. *Water Resources Research*, **54**(10), 8309–8331.

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.