

# **WSHAPE guide**

---

For version 1.0, 19 November 2007

**Alexander Shirokov**

---

Interparticle force implementation for interaction of W-shaped particles

Copyright (C) 2007 Alexander Shirokov

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

# Table of Contents

<b>1</b>	<b>WSHAPE guide</b>	<b>1</b>
1.1	Introduction	1
1.2	C-code Implementation with <code>wshape.c</code> and <code>wshape.h</code>	1
1.3	Generating Analytic Expressions and Plots	3
1.3.1	Raw Expressions with <code>wshape-law-analyt.out</code>	3
1.3.2	Organized Expressions with <code>group.py</code>	4
1.3.3	Paper Plots	5
1.4	Source Code Generators	5
1.4.1	Generating Tables of Coefficients	5
1.4.2	Generating <i>Mathematica</i> Expressions	6



# 1 WSHAPE guide

## 1.1 Introduction

This guide documents version 1.0 of the WSHAPE package, – the source code supplement for astro-ph/0711.2989 (<http://lanl.arxiv.org/abs/0711.2989>) paper (later referred to as the *paper*) on adaptive mass refinement by Alexander Shirokov. The most recent version of the package can be downloaded from <http://www.gracos.org/wshape>.

If you wish to use C-implementation of W-shape profiles you just need C-compiler. GSL library is required for extended Plummer and gaussian force laws. If you wish to reproduce all the results in *paper* including the plots, we also require Python interpreter (our version was “python-2.5-19”), and MATLAB.

Over the course of the following procedures, hundreds of auxiliary files may be generated. Type `make clean` at any time to erase all the extra files.

Note that while required precision has been achieved for the purpose of *paper* further testing may be required if you wish to use unusual parameters. For example, due to polynomial structure of our closed form expressions you may have to deal roundoff effects close to zero separation. We encourage basic testing (e.g. making a plot) before actually using these routines. Please report any bugs or fixes to the authors of this package. Thank you.

## 1.2 C-code Implementation with `wshape.c` and `wshape.h`

Files ‘`wshape.c`’ and ‘`wshape.h`’ provide C-implementation for force and potential laws considered in the *paper* with the following functions. They can be separated from this package used independently.

```

/* Reference laws */

/* Specifies the smoothing scale */
typedef struct wshape_smscale_t {
    float eps;
} wshape_smscale_t;

/* Newtonian potential/force law */
float wshape_law_newton (int law, float r);

/* Plummer potential/force law */
float wshape_law_plummer(int law, wshape_smscale_t *S, float r);

/* Spline (traditional) potential/force law */
float wshape_law_spline(int law, wshape_smscale_t *S, float r);

/* Parameter 'law' takes one of the following values. */
enum {
    /* The return value will be interaction potential law. */

```

```

    WSHAPE_LAW_POTEN,
    /* The return value will be interparticle force law. */
    WSHAPE_LAW_FORCE,
};

/* W-shape particles */

/* Specifies the smoothing scales in a generic pair */
typedef struct wshape_smscales_t {
    wshape_smscale_t particle1, particle2;
} wshape_smscales_t;

/* W-shape density profile, as function of radial position */
float wshape_density (int scaling, wshape_smscale_t *smscale
    , float r, int n);
/* potential/force law for a pair of two general W-shape particles */
float wshape_law_gener (int law, int scaling, wshape_smscales_t *smscales
    , float r, int n);
/* potential/force law for a pair of two identical W-shape particles */
float wshape_law_ident (int law, int scaling, wshape_smscale_t *smscale
    , float r, int n);
/* Potential/force law for a pair including one W-shape and one point particle */
float wshape_law_point (int law, int scaling, wshape_smscale_t *smscale
    , float r, int n);

/* Parameter 'scaling' takes one of the following values. */
enum {
    /* no scaling: the diameter og W-shapes equals epsilon by definition. */
    WSHAPE_SCALING_NONE,
    /* use potential-based scaling of W-shapes: the diameter and
    normalization of W-shape are scaled yo match the plummer potential
    asymptotics at zero separation */
    WSHAPE_SCALING_BY_POTEN,
    /* use force-based scaling of W-shapes: the diameter and
    normalization of W-shape are scaled yo match the plummer force
    asymptotics at zero separation */
    WSHAPE_SCALING_BY_FORCE,
};

/* Maximum allowed value of parameter 'n' */
#define WSHAPE_NMAX    4

/* Raw Hockney-Eastwood window functions */

```

```

float wshape_hockney_eastwood(int n, float s);

/* Symmetric combination \sqrt( (eps1^2+eps2^2)/2 ) motivated by gaussian */
float eps_sym(float eps1, float eps2);

/* The following routines are only available with GSL linked */
#ifdef WITH_GSL
#include "gsl_sf_bessel.h"
#include "gsl_sf_erf.h"

/* Potential/force law in a pair of gaussian clouds of general smoothing */
float wshape_gaussian_law_ident(int law, wshape_smscale_t *S, float r);

/* Plummer density cloud */
float wshape_plummer_density(wshape_smscale_t *S, float r);

/* Potential/force law in a pair of plummer clouds of general smoothing */
float wshape_plummer_law_gener(int law, wshape_smscales_t *S, float r);

/* Potential/force law for a pair including one Plummer cloud and one point particle */
float wshape_plummer_law_point(int law, wshape_smscale_t *S, float r);
#endif

```

## 1.3 Generating Analytic Expressions and Plots

This Section documents a few utilities that call functions in ‘wshape.h’ to produce the plots for the *paper*, or wanted analytic expressions.

### 1.3.1 Raw Expressions with wshape-law-analyt.out

Raw analytic expressions with evaluation are generated with wshape-law-analyt.out as follows. First type

```
cc -o wshape-law-analyt.out wshape.c -lm -D WSHAPE_ANALYT
```

to compile executable executable ./wshape-law-analyt.out whose usage is as follows

```
./wshape-law-analyt.out DISPMODE LAW SCALING N PAIRTYPE SMOO SEPAR
```

where the tokens in the argument list are explained below

#### *DISPMODE*

The allowed values are either ‘DN’ to show the numeric value only, or ‘DY’ to display full analytic expression.

#### *LAW*

The allowed values are ‘potenlaw’ and ‘forcelaw’. This sets for the potential or force law output respectively.

#### *SCALING*

The allowed values are either ‘wbn’ to specify  $W_n$ -shape and  $W$ -shape particles with specified diameter. Use ‘wbn’ and ‘wfn’ for potential- and force- scaled  $W$ -shapes (respectively) with specified smoothing scales.

*N* Specify the smoothness parameter *n*. The allowed values are 1,2,3, and 4.

*PAIRTYPE*

The allowed values are ‘g’ (for interaction of spheres of different diameters or smoothing scale, for force or potential scaled shapes); either ‘p’ (interaction of a particle with a pointlike particle), or ‘i’ (interaction of two identical particles).

*SMOO*

If *PAIRTYPE* equals ‘g’ this parameter equals a pair of column separated floating point numbers.

If *PAIRTYPE* equals ‘p’ or ‘i’, then this parameter is just one number (passing numbers in the format suitable for *PAIRTYPE* equals ‘g’ will result in an error).

If *SCALING* equals ‘wbn’ the passed number/numbers specify the spatial diameter/s of the W-shape particle/s. If *SCALING* equals ‘wsn’ the passed number/s specify the smoothing scale/s of the S-shape/s.

*SEPAR* A floating point number specifying spatial separation in a pair of particles.

As an example, execute the following

```
FLAGS="DY forcелaw wfn 4 g 1.:1. 0.0001"
./wshape-law-analyt.out $FLAGS
```

### 1.3.2 Organized Expressions with group.py

Python executable `group.py` can be used to process the output of `wshape-law-analyt.out` to arrive to a more organized output in which the terms are rearranged into the groups of equal power of separation. These groups appear in order of their fractional contribution to the total value. There is an option to display only those groups whose contribution is limited by a specified fraction of total value.

The usage is as follows

USAGE:

```
group.py [OPTIONS] [FILE]
```

Organize the output of ‘wshape-law-analyt.out’ reading data from standard input; or from FILE, if given.

OPTIONS:

-h

Show the help message and exit.

-l float32

Only display the groups of powers of separation whose combined fractional contribution to the return value exceeds the limit specified.

As an example of usage, execute the following

```

FLAGS="DY forcelaw wfn 4 g 1.:1. 0.0001"

./wshape-law-analyt.out $FLAGS

./wshape-law-analyt.out $FLAGS | ./group.py

./wshape-law-analyt.out $FLAGS | ./group.py -l 1.E-3

```

Executing `./scaling.py` command uses `group.py` to produce the table of force and potential scaling coefficients for the *paper*.

### 1.3.3 Paper Plots

To generate data, for the plots in the *paper*, type:

```

cc -o wshape-data.out wshape.c wshape-tests.c -lm
./wshape-data.out

```

This will however generate only part of the plots. If you have GSL library installed, please either type

```

cc -o wshape-data.out wshape.c wshape-tests.c -lm -D WITH_GSL

```

and follow the hints in error messages, or modify the setting of `GSLPATH` in ‘`Makefile`’ to point out the location of GSL installation on your machine. Then type `./wshape-data.out`.

You should see the following output

```

now using C-math library only..
now using GSL also..

```

The data is now generated. In *MATLAB*, type `paperplots`; this will run the *MATLAB* script ‘`paperplots.m`’ which generates all the plots presented in the *paper*. The output `*.fine.eps` files will appear in the same directory. Getting this data is a good starting point if you wish to find out more information about our profiles.

## 1.4 Source Code Generators

This subsection documents the technique used to generate the static C-arrays in file ‘`wshape.c`’ used in Section 1.3.3 [Paper Plots], page 5 and the tables in the Appendix of the *paper*.

### 1.4.1 Generating Tables of Coefficients

Type

```

./wshape.py potentials

```

to generate tables of potential coefficients. Type

`./wshape.py forces`

to generate tables of force coefficients.

This procedure takes seconds to finish and outputs about 230 new auxiliary text files. The main output files in this pool are `'potentials.c'`, `'potentials.tex'`, `'forces.c'`, and `'forces.tex'` contain tables of coefficients in LaTeX and C-format; the contents of these files were used for the *paper* and filling the static arrays in `'wshape.c'`. The detailed description of what's being done in the `./wshape.py` procedure is outside the scope of this manual; the source files however are rather well commented and will provide all the necessary information.

If you wish to automatically compile the generated files with `latex`, please add the `-l` flag just following the `wshape.py` command.

The large numbers in tables are not reducible. In order to check this you have the prime decomposition output option; just add the `-p` flag following the `wshape.py` command and see the usual output files (e.g. `'potentials.tex'`).

### 1.4.2 Generating *Mathematica* Expressions

Files `'f-n.src'` and `'p-n.src'` (where  $n$  is the integer *smoothness parameter*) are included with this distribution. If your directory has these files, you can go ahead with the procedure in Section 1.4.1 [Generating Tables of Coefficients], page 5.

In order to generate these files in *Mathematica*, open file `'force.nb'` or `'poten.nb'`. Close to the top of the input, set value of the smoothness parameter `n` from the list of currently supported values: 1,2,3, and 4. Press [SHIFT] [ENTER] to run computation in *Mathematica*. Once finished, save the output by clicking in the *Mathematica* Menu Toolbar: `File-> Save as Special-> Text`. Select the output file name. This is how files `'f-n.src'` and `'p-n.src'` were generated in *Mathematica* v. 5.1.