SOWOS: an open-source program for the three-dimensional Wulff construction

Daniele Scopece†

Department of Materials Science, University of Milano-Bicocca, via R. Cozzi 53, I-20125 Milano, Italy.
Correspondence e-mail: daniele.scopece@mater.unimib.it, daniele.scopece.science@gmail.com

A Fortran90 program for the determination of the Wulff construction, starting solely from the directions of the bounding facets (defined by the user), is presented. *SOWOS* stands for solid of Wulff open source, and the program is distributed freely with no charge to the user, being readily available to the community for immediate use. Its simple algorithm (which will be explained) allows the determination of complex solids with hundreds of facets in just seconds on any machine, requiring only a small amount of memory. It is able to determine even the smallest facets and shortest edges and to distinguish almost adjacent vertices. The output files give a complete range of information about the structure: the coordinates of the vertices and the facets common to them, the extension of the facets and bounding vertices, and the length of the edges and extreme vertices. These details enable the reconstruction of the shape in any other (commercial) software for further processing. Visualization is straightforward *via* the free program *gnuplot*. A feature for the creation of cubic crystal atomistic models of the resultant solids is included. The program may be a useful tool for crystallography, nanostructures and any other field where crystal facets are involved.

© 2013 International Union of Crystallography
Printed in Singapore – all rights reserved

1. Introduction

Since the very dawn of the study of condensed matter, the determination of crystal structure has played a major role. In the past couple of decades this interest has been greatly renewed, owing to the field of nanostructures and heteroepitaxial islands (Medeiros-Ribeiro *et al.*, 1998; Robinson *et al.*, 2009).

The shape of crystallites is a fundamental ingredient (among others) in determining relevant physical quantities, such as the energy of the facets (Stekolnikov & Bechstedt, 2005), the step-formation energy (Giesen *et al.*, 2001; Bonzel & Nowicki, 2004), the velocity of facet growth (Sun *et al.*, 2008, 2011; Romanyuk *et al.*, 2007) or other physical effects (Kim *et al.*, 2009). Crystallite shape can, in turn, be determined using the Wulff theorem and construction (Wulff, 1901; Rusanov, 2005, 1996; Longo *et al.*, 2011; Falub *et al.*, 2012).

Although a large number of programs are available to create this construction, most of them either are no longer developed or supported (Wulffman Working Group, 2012) or lack the versatility needed for a complete analysis of the resulting geometry, such as knowledge of the areas of single facets, hence the need for an easy-to-use, fast, flexible and free tool to obtain this geometry. The condition of versatility arises because the code has to be suitable to analyse both free crystallites and crystallites constrained on a substrate, as in the case of Ge/Si, InAs/GaAs (Costantini *et al.*, 2005) or any heteroepitaxial island.

The Fortran90 code presented here is designed to be run in Linux environments and it is made freely available for download ready for immediate use. It allows the user to fulfil this task on any computer and provides all the information necessary to recreate the solid, ready

for use in any other software or hand-made code for further analysis. Some examples of the geometries built using *SOWOS* are reported in Fig. 1.

The code is written in a modular fashion with commented subroutines, to allow it to be easily understood and improved by the user. Modifications are possible and encouraged, provided that any improvements are communicated to the author in order to make them available to the community (see §4 below for the licence agreement). To this end, what follows is a detailed description of the basic concepts and definitions of the Wulff solid adopted, the simple

Figure 1

Examples of solids obtained using *SOWOS* (see text). (a) A top view of a {137} island observed in InAs/GaAs heteroepitaxy (Costantini *et al.*, 2005). (b) A side view of the Ge/Si barn (Zhang *et al.*, 2010). (c) A side view of a fictitious island with 169 facets. (d) A side view of a pencil-shaped crystal. (e) A side view of a diamond-shaped crystal. (f) The top view of (c).

† Present address: Empa, Swiss Federal Laboratories for Materials Science and Technology, Überlandstrasse 129, 8600 Dübendorf, Switzerland.

By definition, the Wulff construction is the convex envelope of the Wulff γ plot containing a fixed point, conventionally called the Wulff point (Rusanov, 1996). *SOWOS* neglects the polar γ plot and focuses on the creation of the convex solid, starting from the Cartesian coordinates of the normals $\mathbf{n}_j$ to the planes specified in the input file only and neglecting all other directions. The distance of the j th plane from the Wulff point (defined as the origin of the axes in *SOWOS*) is denoted w_j for the reasons outlined below. Each plane divides the three-dimensional space into two halves (see Fig. 2). In the following, a surface (or facet) is defined as S type if the Wulff solid lies in the half-space containing the Wulff point; it is said to be B type otherwise. This distinction is made to allow the solid to exclude the Wulff point for general purposes, and also to allow the modelling of hetero-epitaxial islands lying on any substrate direction (B stands for the direction of the base of the island and S for exposed surfaces). This allows one to have a model of an island lying on any substrate (Robinson *et al.*, 2009; Sanduijvat *et al.*, 2012) and to define important parameters of the islands (see §3.7 below).

Any facet is therefore identified by a vector $F_j = [T_j, \mathbf{n}_j, w_j]$, where T_j is the type, S or B. Since these planes are the borders of the solid, it is necessary to know the condition for a point $\mathbf{r}$ to lie on the plane F_j . This condition is conveniently expressed as (here $\hat{\mathbf{n}}_j$ is a versor)

$$\Rightarrow$$

$$\mathbf{r} \cdot \mathbf{n}_i = w_i |\mathbf{n}_i|. \quad (2)$$

This code can be used to replicate the geometry of crystallites or heteroepitaxial islands by inserting the directions $\mathbf{n}_i$ of the revealed facets (Rastelli & Känel, 2002) and by ‘tuning’ the w_j values (in fact their relative ratios) to recreate manually the observed geometry [*i.e.* the ratio between facet extensions, as adopted by Sanduijav *et al.* (2012) and Zhang *et al.* (2010)]. This allows one to quantify the (relative) extensions of the facets in complex multi-faceted islands

Figure 2
Definition of the Wulff solid adopted in *SOWOS*. The Wulff point is at the origin of the axes. Black arrows are the normals $w_j \hat{n}_j$ to the facets F_j . F_1 is a B type and F_2 an S type, and hence the solid is in the region included between these two planes and in this example it does not include the Wulff point. Of the points $\mathbf{r}_1$, $\mathbf{r}_2$ and $\mathbf{r}_3$ (circles), only the first one belongs to the solid. The triangles and squares are the ‘projections’ discussed in §3.3 and are used to deduce the criteria of Table 2 to check whether a point is inside the solid.

(which are difficult to quantify experimentally) once the actual geometry observed, *e.g.* via STM measurements, is obtained. It is suggested that one starts with all $w_j = 1$ (or 10 or any other value) and then modifies them one by one until the sought geometry is obtained. Here, the symbol w is used instead of the conventionally adopted γ to avoid confusion. In strained nano-islands (such as Ge/Si or InAs/GaAs), the surface energy of the facets (γ) (Tersoff & Tromp, 1993; Fichthorn *et al.*, 2011) depends on the strain state of the facet itself (Lu *et al.*, 2005; Scopece *et al.*, 2012; Migas *et al.*, 2004; Chen *et al.*, 2012), which is in turn a function of the island shape and can assume a different value from that needed in the Wulff theorem to recreate the geometry.

Here, the algorithm employed in version 2.0 of *SOWOS* is described. Version 1.0, using a less efficient method, is ignored.

SOWOS automatically reads the file `Input_SOWOS.input`. Here the name of the file containing the list of facets F_j to be considered is listed. Let us call this file `Wulff.input`. This file must be written by the user according to his/her needs (*SOWOS* takes it as an input) and must have this structure:

```

2
B 0 0 1      w  0.0      x 1
S 1 3 7      w 10.0      x 0

```

The first line contains the total number of lines to be read. The following lines contain, in order, the type T_j , the normal $\mathbf{n}_j$ (the Cartesian components of which are expressed as integer numbers), the value of w_j and a numerical value after the x . This last value is defined as the multiplicity M_j of the j th facet. It states how many facets are considered in the code whose normal $\mathbf{n}_j$ creates the same (polar) angle θ with the [001] direction (defined as the z axis) as the angle of the corresponding entry in the input file according to the symmetry of the crystal. Since *SOWOS* was originally developed to model islands in cubic crystals (Ge/Si and InAs/GaAs, to mention just two), the multiplicity defined here follows cubic symmetry. The normals considered for the different cases are reported in Table 1. The case ' $x - 1$ ' is included to allow full three-dimensional structures, since in this case the third index is also changed in sign. By properly setting these values in the input file the user can impose any symmetry to be considered in *SOWOS*.

SOWOS reads these multiplicities and exports them into the file `out.normals-multiplicity.txt`, which has the same structure as `Wulff.input` and where the labels of the facets that are used throughout the algorithm are reported as well. Panels (a) and (b) of Fig. 3 reproduce the different islands for {137} pyramids on 001 in the case of $M_{\{137\}} = 4$ or 0, respectively.

Figure 3
Examples of the different geometries obtained for different multiplicities for a {137} pyramid on (001), viewed from above. (a) The $\times 4$ case and (b) the $\times 0$ case.

It is worth highlighting that the possibility of $M_j = 1$ enables *SOWOS* to be very versatile, making it suitable for any crystal structure other than cubic (once the proper directions have been defined) or when not all the facets dictated by the crystal symmetry appear in the structure [see, for example, Fig. 1(a) (Kiravittaya *et al.*, 2006; Costantini *et al.*, 2005)].

The w values can be written in the input file or, alternatively, can be computed by *SOWOS* as the reciprocal of the modulus of the vector starting from the indices stated. This latter case can be imposed by setting the variable `l_modulus_normals` equal to 1 in `Input_SOWOS.input` (it is set to 0 for the former case), in which case all the w values are multiplied by a common factor given by `factor_modulus` as stated in `Input_SOWOS.input`. In this situation, the file `Wulff.input` is overwritten to keep track of the actual values adopted in the calculation.

All the output files from *SOWOS* have the name `out.*` and are text files, in order to be easily readable by any text editor or as input files for any other program.

The run of *SOWOS* for the input file just reported (137×0) is as follows:

```
*****
WELCOME TO
SOWOS
= Solid Of Wulff, Open Source
A code for the WULFF CONSTRUCTION in 3D

Version SOWOS.v02.00

written by Daniele Scopece
daniele.scopece.science@gmail.com
*****

Everything is read from the file <<Input_SOWOS.input>>

--> Input file for facet is <<Wulff.input>>

[...]

-----
FINISHED FACET: 1 / 9 {times_inside = 24 }
FINISHED FACET: 2 / 9 {times_inside = 45 }
FINISHED FACET: 3 / 9 {times_inside = 60 }
FINISHED FACET: 4 / 9 {times_inside = 70 }
FINISHED FACET: 5 / 9 {times_inside = 76 }
FINISHED FACET: 6 / 9 {times_inside = 79 }
FINISHED FACET: 7 / 9 {times_inside = 80 }
FINISHED FACET: 8 / 9 {times_inside = 80 }
FINISHED FACET: 9 / 9 {times_inside = 80 }
-----

>>>> [List of output files ...]
+++++
NORMAL END of the code
THE END
```

The procedure followed by the algorithm and the corresponding output files are described below.

3.2. Determination of vertices

In a generic solid a vertex point is common to at least three distinct planes. The coordinates of this point $\mathbf{r} = [x, y, z]$ must satisfy equation (2) for the three planes characterized by the set of normals and w equal to $(w_1, \mathbf{n}_1)$, $(w_2, \mathbf{n}_2)$ and $(w_3, \mathbf{n}_3)$:

$$\begin{cases} \mathbf{r} \cdot \mathbf{n}_1 = w_1 |\mathbf{n}_1| \equiv C_1, \\ \mathbf{r} \cdot \mathbf{n}_2 = w_2 |\mathbf{n}_2| \equiv C_2, \\ \mathbf{r} \cdot \mathbf{n}_3 = w_3 |\mathbf{n}_3| \equiv C_3. \end{cases} \quad (3)$$

This system has a unique solution (*i.e.* a point), provided that the matrix of the indices n_{ij} is invertible. The solution is

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} n_{11} & n_{12} & n_{13} \\ n_{21} & n_{22} & n_{23} \\ n_{31} & n_{32} & n_{33} \end{bmatrix}^{-1} \begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix}. \quad (4)$$

Table 1

The normals considered for different multiplicities in the input file `Wulff.input` for a generic normal (1,2,3).

In the code, repeated facets are considered just once. Commas are inserted and the minus sign is specified for visual clarity.

Multiplicity M_j	Normals considered in the solid built by <i>SOWOS</i>
x 1	(1,2,3)
x 2	(1,2,3) + (-1,-2,3)
x 4	(1,2,3) + (-1,-2,3) + (-1,2,3) + (1,-2,3)
x 0	(1,2,3) + (-1,-2,3) + (-1,2,3) + (1,-2,3) + (2,1,3) + (-2,-1,3) + (-2,1,3) + (2,-1,3)
x -1	(1,2,3) + (-1,-2,3) + (-1,2,3) + (1,-2,3) + (2,1,3) + (-2,-1,3) + (-2,1,3) + (2,-1,3) + (1,2,-3) + (-1,2,-3) + (-1,-2,-3) + (1,-2,-3) + (2,1,-3) + (-2,1,-3) + (2,-1,-3)

Table 2

Criteria for a generic point $\mathbf{r} = [x, y, z]$ to be outside the solid, as explained in §3.3 and deduced from Fig. 2.

The assumption is made that $n_{j3} \neq 0$, hence a point r_{pj} on the facet has coordinates $[x, y, z_{pj}]$. If $n_{j3} = 0$, the same criteria apply to the x (y) coordinate, as long as n_{j1} (n_{j2}) $\neq 0$.

Type of facet j	Sign of (n_{j3})	Condition to be outside
S	$n_{j3} > 0$	$z > z_{pj}$
S	$n_{j3} < 0$	$z < z_{pj}$
B	$n_{j3} > 0$	$z < z_{pj}$
B	$n_{j3} < 0$	$z > z_{pj}$

Any combination of three planes is considered in *SOWOS*, the number of possibilities being equal to

$$\binom{N}{3},$$

where N is the total number of distinct planes. The combination of facet i with any two other facets is analysed in the part entitled `FINISHED FACET` in the run (see §3.1).

Several tests on *SOWOS* have shown that this method allows distinction between points even when they are very close to each other (up to 0.001 units apart if double-precision real variables are used). However, when the resultant solid is too small some facets or edges cannot be identified. If this is the case, it is sufficient to scale up all the w_j values in the input file to a common factor, thus leaving the shape unaltered while enlarging the volume.

SOWOS saves a list of these points in a temporary file.

3.3. Vertices (points) inside the solid

The vertices determined in the previous section are the class of all possible intersections. We must now determine whether these points are inside the Wulff solid. By definition, a given point is outside the solid if it is outside at least one of the half-spaces delimited by the vectors $w_j \hat{\mathbf{n}}_j$. It is thus convenient to check a condition for the point to be outside the solid.

Taking Fig. 2 as a reference, we must determine a condition for which the points $\mathbf{r} = [x, y, z]$ (circles in the figure) belong to the solid.

Let us define the point $\mathbf{r}_{pj}$ as a point with the same x and y as point $\mathbf{r}$ but which lies on facet F_j . It has coordinates $[x, y, z_{pj}]$, with

$$z_{pj} = \frac{w_j |\mathbf{n}_j| - x n_{j1} - y n_{j2}}{n_{j3}}. \quad (5)$$

These 'projections' are represented by squares or triangles for facets F_1 and F_2 , respectively, in Fig. 2. By inspecting that figure, we deduce the conditions for the point $\mathbf{r}$ to be outside the solid, reported in Table 2 for S- or B-type facets. If these criteria are met by at least one facet, the point does not belong to the solid.

In the case where $n_{j3} = 0$, such as (110), similar conditions can be used for the other coordinates (x , x_{pj} or y , y_{pj}).

This is the bottleneck of the code, but since the number of points is limited this function is called a limited number of times (called `times_inside` in the run shown in §3.1), enabling *SOWOS* to construct a geometry with ~200 facets (Figs. 1c and 1f) in ~15 s on a laptop with an AMD Athlon II P320 dual-core processor running at 800 MHz.

Since a vertex can be in common with more than three planes, in the list of vertices the same vertex can appear more than once. The list is therefore checked for repeated points and the final vertices inside the solid are reported in the file `out.vertices.dat`.

3.4. Determination of edges

After obtaining the vertices, we must determine which pairs of them are bound together in the solid to form the edges. The vertices connected together are those that have at least two facets in common. The coordinates of the edges are written in the file `out.gnuplot.xyz` to reproduce the solid in *gnuplot* (Williams & Kelley, 2008). In this file the following data are reported in order: the x , y and z coordinates of the vertex, the label of the vertex, and the labels of the planes that create the edge made by two successive points. Figs. 1, 3 and 4 were produced with this file by loading `out.plot-gnuplot.plt` into *gnuplot*.

The lengths of the edges are listed in the file `out.edges_length.dat`, along with the labels of the facets that create the edge and the direction in space of the edges themselves.

3.5. Calculation of areas

For practical purposes it is important to know the area of each facet. *SOWOS* determines this from the coordinates of the vertices bounding the facet, using the triangulation method shown in Fig. 4 (red lines), where two vertices are next to each other (*i.e.* those creating an edge) and the third is the baricentre of the facet ($\mathbf{B}_F$ in the figure). In this way the triangles do not cross each other and the area of the facet F_j is the sum of the areas of the triangles, which are given as a function of the semiperimeter $p = (a + b + c)/2$ of each triangle:

$$S_j = \sum_k^{\text{triangles}} s_k, \quad (6)$$

Figure 4
Calculation of area and volume with triangulation in the case of a half-octahedron. $\mathbf{B}_F$ is the baricentre of the (111) facet. Red lines delimit the triangles for estimation of the facet area (§3.5). $\mathbf{B}_S$ is the baricentre of the entire solid, *i.e.* the apex of the pyramids whose bases are the facets for the calculation of the volume of the solid (§3.6).

$$s_k = [p(p-a)(p-b)(p-c)]^{1/2}. \quad (7)$$

This is a generic formula valid for any shape of facet, not only regular polygons.

A list of the facet labels, the normals, the area, the coordinates of the baricentres and the labels of the bounding vertices are reported in the file `out.AREAS.dat`. The triangulation can be plotted in *gnuplot* via the file `out.plot-gnuplot.plt` by uncommenting the line containing `out.gnuplot-triangulation-facets.xyz`.

3.6. Calculation of volume

The same triangulation procedure can be used to determine the volume of the solid. In this case the solid is divided into small pyramids, for which the bases are the facets and the apex vertex is the baricentre $\mathbf{B}_S$ of the entire solid (see Fig. 4, blue lines, for an example). The height of the pyramid whose base is the facet F_j is given by the distance of the point $\mathbf{B}_S$ from that facet:

$$h_j = |w_j - \mathbf{B}_S \cdot \hat{\mathbf{n}}_j|. \quad (8)$$

Since we are handling convex solids, the pyramids do not overlap with each other and the volume of the entire solid is given by

$$V_{\text{tot}} = \sum_j^{\text{pyramids}} v_j, \quad (9)$$

$$v_j = S_j h_j / 3. \quad (10)$$

The list of volumes v_j and values h_j is reported in the file `out.volume.dat`, along with the total volume and the coordinates of the baricentre $\mathbf{B}_S$.

3.7. Useful parameters for heteroepitaxy

In heteroepitaxy, the shape of an island is characterized by the aspect ratio (AR), defined as the height of the island divided by the square root of its base (Zhang *et al.*, 2010). *SOWOS* takes as the base the first B-type facet encountered in the `Wulff.input` file. This is the historical reason why B-type surfaces are introduced in *SOWOS*. The orthogonal distances of the vertices from the base are reported in `out.distance-vertices-B.dat`. The height of the island is taken as the maximum of these values. The label of the vertex with the maximum height, the base area and the corresponding value of the AR are reported in `out.AR.dat`.

Other parameters widely employed in multiscale modelling of heteroepitaxy are the facet area divided by the volume of the entire crystal to the power of 2/3: $\beta_j \equiv S_j/V_{\text{tot}}^{2/3}$ (Chen *et al.*, 2012; Scopece *et al.*, 2012; Scopece & Beck, 2013). These values are pure numbers and are independent of the volume of the solid. The values for each facet are listed in `out.beta_facets.dat`. Since facets belonging to the same multiplicity group very often have the same surface energy, the β values are re-grouped in the file `out.beta_groups.dat`. Comparison of the β values gives a direct comparison of the relative abundance of the facets in the crystal and this parameter may turn out to be useful in crystallography as well. Because of the properties of the Wulff construction, facets with higher w_j will have a lower β_j , vanishing for a high enough value of w_j .

Similar adimensional factors for the edges should be computed. This feature will be introduced in a future version of *SOWOS*.

3.8. Atomistic solids

In some circumstances, having an atomistic model of the crystal or island is valuable (Retford *et al.*, 2007; Hamilton, 2006; Zhou & Fichthorn, 2012). This feature is included in *SOWOS* for the

Figure 5

Examples of the atomistic island and substrate obtained with *SOWOS*: diamond on a (108) substrate. (a) Plot in *gnuplot*. (b) Visualization in *VESTA* (Momma & Izumi, 2008). (c) and (d) Enlargement of the *VESTA* plot, showing the atomic steps and matching of the substrate. *SOWOS* can match the substrate at any orientation (see §3.8).

modelling of cubic crystals (simple cubic, body-centred, face-centred or diamond), where the same criteria as in §3.3 and Table 2 are valid for the atoms to be inside the solid.

The lattice constants can be tuned at will. The file created is `out.atomistic-cell.cls` in the `clsman` type, but it can be manipulated by the user to convert it into any cell type. The cell can be viewed in *gnuplot* using `out.atomistic-plot.plt`.

In *SOWOS* the geometry is such that the axes correspond to the directions [100], [010] and [001] for x , y and z , respectively. Since the substrate is a convex solid as well (typically a parallelepiped), it can also be modelled by the code, as shown in Fig. 5. As is evident from this figure, the atomic structure of the facets (steps and symmetry) appears automatically. Care must be taken to erase some atoms one by one at the very border, since they might appear in both the island and the substrate.

In this way, an atomistic model for the solid is created that can be analysed with some semiempirical potential (Fichthorn *et al.*, 2011; Tersoff, 1986, 1990) or be handled to include line defects such as dislocations.

3.9. Importing into other software

Since all the information available from the `out.*` files completely defines the solid, it can be manipulated to create any format in order to define the solid in any program.

An example in this respect is the creation of a three-dimensional geometry to be imported into *COMSOL Multiphysics* (COMSOL, 2012). A 'brute-force' method of importing this geometry is to create a mesh in *COMSOL*, starting from the vertices of the solid. To improve the precision the baricentres of the facets, $\mathbf{B}_F$, and the average point of each edge are reported in the file `out.solid_by_points.xyz`. The mesh is created using `COMSOL_Import_geometry_SOWOS_v04.m` and the solid is created in Version 3.5 of *COMSOL* with a *MATLAB* (The MathWorks Inc., Natick, MA, USA) interface. An example of the obtained geometry is reported in Fig. 6 with 336 facets. This method has the shortcoming of losing the smallest facets that are revealed by *SOWOS*, so an improved method will be introduced in future versions of *SOWOS*.

3.10. Euler rule

In some cases the facets introduced in the input file cannot create a closed solid [as is the case for Ge {105} pyramids on an Si(1 1 10)

substrate; Chen *et al.* (2012)]. To monitor such a possibility, the Euler rule for convex solids (number of facets – number of edges + number of vertices = 2) is checked in any calculations. However, since this rule is a necessary condition for the solid to be convex, the final geometry should always be checked graphically using `out.plot-gnuplot.plt`.

4. Distribution and documentation

SOWOS is distributed 'as is' without any warranty and it is freely downloadable from the URL <http://www.danielescopece84.altervista.org/sowos.html>. Alternatively, it can be requested from the author by sending an email with the subject 'SOWOS'.

The software may be modified provided that the modifications are communicated to the licence holder in order to make them available to the user community. The licence conditions are reported in the file `000-README__License-SOWOS.v01.txt`. Any work using *SOWOS* or part of it should cite the present publication. Requests for support or on other matters may be directed to the author by email.

Figure 6

The geometry created with *SOWOS*, imported into *COMSOL Multiphysics* (336 facets) via the mesh method explained in §3.9. This is an example of the possibility of creating non-symmetric shapes in *SOWOS* (and hence of its versatility). The upward facets have the same Miller indices as the downward ones (with the third index opposite in sign), but their w_i values differ by a factor of two, thus creating an elongated shape.

SOWOS requires two free programs to be run: *gfortran* (Version 4.4.3 or higher), and *gnuplot* (Version 4.2 or higher) to visualize the structure. In the tarball are some examples of input files (which play the role of a mini-tutorial) and an instruction file.

5. Future developments

Future versions of *SOWOS* will tend towards greater generalization, such as to allow the placement of the Wulff point away from the origin of the axes, and to allow the change of w_j with time in order to observe the kinetic evolution of a Wulff construction. New crystal geometries will be introduced for the construction of atomistic solids (zinc blende, hexagonal close-packed structures *etc.*). A graphical user interface may be developed in the future. Specific requests from users can also be implemented and are welcome.

6. Conclusions

An open-source program has been presented for the creation of a three-dimensional Wulff construction from a user-specified list of the facets to be considered.

No special technical or programming skills are required, since the algorithm is straightforward and very linear, so this code is made available to the community of crystallographers for immediate application and exploitation of this fruitful concept of the Wulff construction, to save others the time and effort of writing it from scratch.

It may be a useful instrument for both crystallographers and researchers studying self-assembled islands.

Dr Guglielmo Vastola is acknowledged for his invaluable help in introducing the author to the concept of the Wulff construction and for explaining the meaning of the β factors, not to mention for his early pioneering work on this concept in our group. Dr Riccardo Gatti and Dr Francesca Boioli are acknowledged for their invaluable support on the procedure to import the final geometry into *COMSOL*.

References

- Bonzel, H. & Nowicki, M. (2004). *Phys. Rev. B*, **70**, 245430.
Chen, G., Sanduivaj, B., Matei, D., Springholz, G., Scopece, D., Beck, M. J., Montalenti, F. & Miglio, L. (2012). *Phys. Rev. Lett.* **108**, 055503.

- COMSOL (2012). *COMSOL Multiphysics User's Guide Manual*, <http://www.comsol.com>.
Costantini, G., Rastelli, A., Manzano, C., Acosta-Diaz, P., Katsaros, G., Songmuang, R., Schmidt, O. G., Kanel, H. V. & Kern, K. (2005). *J. Cryst. Growth*, **278**, 38–45.
Falub, C. V., von Känel, H., Isa, F., Bergamaschini, R., Marzegalli, A., Chrastina, D., Isella, G., Müller, E., Niedermann, P. & Miglio, L. (2012). *Science*, **335**, 1330–1334.
Fichthorn, K. A., Tiwary, Y., Hammerschmidt, T., Kratzer, P. & Scheffler, M. (2011). *Phys. Rev. B*, **83**, 195328.
Giesen, M., Steimer, C. & Ibach, H. (2001). *Surf. Sci.* **471**, 80–100.
Hamilton, J. (2006). *Phys. Rev. B*, **73**, 125447.
Kim, K. C., Dai, B., Johnson, J. K. & Sholl, D. S. (2009). *Nanotechnology*, **20**, 204001.
Kiravittaya, S., Songmuang, R., Rastelli, A., Heidemeyer, H. & Schmidt, O. G. (2006). *Nanoscale Res. Lett.* **1**, 1–10.
Lu, G.-H., Cuma, M. & Liu, F. (2005). *Phys. Rev. B*, **72**, 125415.
Longo, V. M., Gracia, L., Stroppa, D. G., Cavalcante, S., Orlandi, M., Ramirez, J., Leite, E. R., Andr, J. & Beltr, A. (2011). *J. Phys. Chem. C*, **115**, 20113–20119.
Medeiros-Ribeiro, G., Bratkovski, A. M., Kamins, T. I., Ohlberg, D. A. A. & Williams, R. S. (1998). *Science*, **279**, 353–355.
Migas, D. B., Cereda, S., Montalenti, F. & Miglio, L. (2004). *Surf. Sci.* **556**, 121–128.
Momma, K. & Izumi, F. (2008). *J. Appl. Cryst.* **41**, 653–658.
Rastelli, A. & von Känel, H. (2002). *Surf. Sci. Lett.* **515**, L493–L498.
Retford, C. M., Asta, M., Miksis, M. J., Voorhees, P. W. & Webb, E. (2007). *Phys. Rev. B*, **75**, 1–8.
Robinson, J. T., Rastelli, A., Schmidt, O. G. & Dubon, O. D. (2009). *Nanotechnology*, **20**, 085708.
Romanyuk, K., Cherepanov, V. & Voigtlander, B. (2007). *Phys. Rev. Lett.* **99**, 126103.
Rusanov, A. I. (1996). *Surf. Sci. Rep.* **23**, 173–247.
Rusanov, A. I. (2005). *Surf. Sci. Rep.* **58**, 111–239.
Sanduivaj, B., Scopece, D., Matei, D., Chen, G., Schäffler, F., Miglio, L. & Springholz, G. (2012). *Phys. Rev. Lett.* **109**, 025505.
Scopece, D. & Beck, M. J. (2013). *Phys. Rev. B*, **87**, 155310.
Scopece, D., Montalenti, F. & Beck, M. J. (2012). *Phys. Rev. B*, **85**, 085312.
Stekolnikov, A. A. & Bechstedt, F. (2005). *Phys. Rev. B*, **72**, 125326.
Sun, Q., Yerino, C. D., Ko, T. S., Cho, Y. S., Lee, I.-h., Han, J. & Coltrin, M. E. (2008). *J. Appl. Phys.* **104**, 093523.
Sun, Q., Yerino, C. D., Leung, B., Han, J., Coltrin, M. E., Sun, Q., Yerino, C. D., Leung, B. & Han, J. (2011). *J. Appl. Phys.* **110**, 053517.
Tersoff, J. (1986). *Phys. Rev. Lett.* **56**, 632–635.
Tersoff, J. (1990). *Phys. Rev. B*, **41**, 3248.
Tersoff, J. & Tromp, R. M. (1993). *Phys. Rev. Lett.* **70**, 2782–2785.
Williams, T. & Kelley, C. (2008). *gnuplot*, <http://www.gnuplot.info/>.
Wulff, G. (1901). *Z. Kristallogr.* **34**, 449–530.
Wulffman Working Group (2012). *WULFFMAN: an Interactive Crystal Shape Constructor*, <http://www.ctcms.nist.gov/wulffman/>.
Zhang, J. J., Montalenti, F., Rastelli, A., Hrauda, N., Scopece, D., Groiss, H., Stangl, J., Pezzoli, F., Schäffler, F., Schmidt, O. G., Miglio, L. & Bauer, G. (2010). *Phys. Rev. Lett.* **105**, 166102.
Zhou, Y. & Fichthorn, K. A. (2012). *J. Phys. Chem. C*, **116**, 8314–8321.