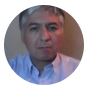
C library for topological study of the electronic charge density

ARTICLE *in* JOURNAL OF COMPUTATIONAL CHEMISTRY · DECEMBER 2012

Impact Factor: 3.59 · DOI: 10.1002/jcc.23083 · Source: PubMed



David Vega
Universidad de Carabobo, UC
17 PUBLICATIONS **211** CITATIONS

Yosslen Aray
Venezuelan Institute for Scientific Research
66 PUBLICATIONS **728** CITATIONS

C Library for Topological Study of the Electronic Charge Density

David Vega,*[a] Yosslen Aray,[b] and Jesús Rodríguez[b]

The topological study of the electronic charge density is useful to obtain information about the kinds of bonds (ionic or covalent) and the atom charges on a molecule or crystal. For this study, it is necessary to calculate, at every space point, the electronic density and its electronic density derivatives values up to second order. In this work, a grid-based method for these calculations is described. The library, implemented for three dimensions, is based on a multidimensional Lagrange interpolation in a regular grid; by differentiating the resulting polynomial, the gradient vector, the Hessian matrix and the Laplacian formulas were obtained for every space point. More complex functions such as the Newton–Raphson method (to find the critical points, where the gradient is null) and the Cash–Karp Runge–Kutta method (used to make the gradient paths) were programmed. As in some crystals, the unit cell has angles different from 90°, the described library includes linear transformations to correct the gradient and Hessian when the grid is distorted (inclined). Functions were also developed to handle grid containing files (grd from DMol® program, CUBE from Gaussian® program and CHGCAR from VASP® program). Each one of these files contains the data for a molecular or crystal electronic property (such as charge density, spin density, electrostatic potential, and others) in a three-dimensional (3D) grid. The library can be adapted to make the topological study in any regular 3D grid by modifying the code of these functions. © 2012 Wiley Periodicals, Inc.

DOI: 10.1002/jcc.23083

Introduction

The "*quantum theory of atoms in molecules*" (QTAIM) developed by Bader et al.[1–8] is very useful to obtain the chemical information from the charge density. QTAIM is a firm, rigorous, and quantum mechanically well-defined theory based on observables such as the electron density or energy density fields. Most modern theories of bonding are based, in one way or another, on the partition of charge (or electronic density) among the different nuclear centers under study, usually according to Mulliken, that is, projected density of states in solid-analysis. In this way, an important amount of the interpretative models of chemical behavior are based on concepts that are known to be poorly defined and giving answers extremely dependent on a whole hierarchy of approximations.[6] QTAIM provides a quantitative link between the total electron density (regardless of how it was generated: calculated or experimental) and important physical properties of a molecule, bypassing the wave function in the analysis. In contrast, QTAIM is a methodology independent from the orbital concept. In particular, it provides a rigorous definition of chemical bond and geometrical structure for all types of molecules and solids and it has proven to be useful in the analysis of physical properties of insulators, pure metals, and alloys.[4–6] High-quality experimental densities of minerals,[9,10] covalent,[11] metallic,[12] and molecular crystals[13,14] have been analyzed in terms of QTAIM concepts. Furthermore, QTAIM-calculations on simple metals,[15] alloys, and intermetallic phases[16,17] have also been reported as well as on molecular,[18,19] covalent, and ionic crystals.[6,11] Software packages that calculate and draw gradient path, laplacian, and density isolines has been published,[20–28] using similar computational methods; however, only few of them are open source code (Aimpac,[25] Multiwfn,[26] DGrid,[27] and Critic[28]).

The topological study consists in determining and characterizing critical points, bond paths, zero flux surfaces, gradient maps, and atomic basins. The critical points are ones where the gradient is null (among these points exist local minima and maxima, and saddle points) and are fundamental to the topological study. The local maxima usually correspond to the atomic nuclei positions, the local minima are cage critical points; and there are two kinds of saddle points, minimum in one direction and maximum in the perpendicular plane (bond critical points), and maximum in one direction and minimum in the perpendicular plane (ring critical points). A bond and interatomic surface representations are obtained by drawing gradient paths close to a bond critical point. To find these points, it is necessary to obtain the gradient vector, $\mathbf{g}(\mathbf{r})$ and the Hessian matrix, $\mathbf{H}(\mathbf{r})$ in any space point $\mathbf{r}$ (using the Newton–Raphson method, later described).

[a] *D. Vega*
Dpto. de Química, Facultad de Ciencias y Tecnología, Universidad de Carabobo, Ciudad Universitaria, Bárbula, Valencia, Venezuela
E-mail: dvega@uc.edu.ve

[b] *Y. Aray, J. Rodríguez*
Centro de Química, IVIC, Apartado 21827, Caracas 1020 A, Venezuela

Contract/grant sponsor: Fondo Nacional de Ciencias y Tecnología (Fonacit) of Venezuela; Contract/grant number: G-2005000424.

$$\mathbf{r}=\begin{bmatrix}x\\y\\z\end{bmatrix}\quad \mathbf{g}(\mathbf{r})=\nabla f(\mathbf{r})=\begin{bmatrix}\frac{\partial f}{\partial x}\\ \frac{\partial f}{\partial y}\\ \frac{\partial f}{\partial z}\end{bmatrix}\quad \mathbf{H}(\mathbf{r})=\begin{bmatrix}\frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x\partial y} & \frac{\partial^2 f}{\partial x\partial z}\\ \frac{\partial^2 f}{\partial x\partial y} & \frac{\partial^2 f}{\partial y^2} & \frac{\partial^2 f}{\partial y\partial z}\\ \frac{\partial^2 f}{\partial x\partial z} & \frac{\partial^2 f}{\partial y\partial z} & \frac{\partial^2 f}{\partial z^2}\end{bmatrix} \tag{1}$$

To replace gradient and Hessian calculation routines in software packages for finding critical points and their properties, the library began to build at early 2000s using Fortran 77. For developing our own programs comprehensively, later on, the library was translated to C language. This C library content was early reported,[29] not in detail, but including some calculation examples. Also, it has been used as a tool in studies of ours group.[30] In this work, we describe exhaustively the C library, providing the source code, method of use and some additional functions for reading grids in different formats.

The library contains functions to calculate the gradient vector and the Hessian matrix, and other functions useful for the topological study.

Approximation to the Function and Derivatives

Starting with the Lagrange polynomial approximation formula,[31,32] eq. (2):

$$P(x)=\sum_{k=0}^{n-1}\frac{\prod_{j=0,\,j\neq k}^{n-1}(x-x_j)}{\prod_{j=0,\,j\neq k}^{n-1}(x_k-x_j)}f_k \tag{2}$$

where n is the number of points (x_j, f_j) used in the interpolation and $P(x)$ is a polynomial of degree $n-1$ that passes through the

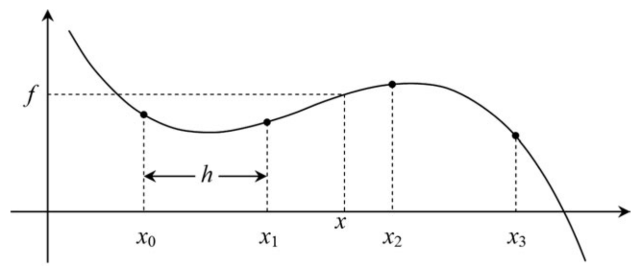


Figure 1. The Lagrange polynomial passes through the points (black dots). When the polynomial is evaluated in another point (x), an approximate value (f) of the function is obtained. When possible, in a piecewise interpolation, it is convenient to choose the points array such that the x value is inside the central interval.[29]

n points. If the x_j values are equally spaced[29] (Fig. 1), then $x_j = x_0 + j\,h$, where h is the distance among the points.

Defining $s = (x - x_\alpha)/h$, with a constant α index (it is convenient to choose α such that x_α and $x_{\alpha+1}$ are the central points of the array: n is even and $\alpha = n/2-1$), solving for x ($x = x_\alpha + s{\cdot}h$) and substituting in the Lagrange polynomial, eq. (2):

$$P=\sum_{k=0}^{n-1}w_{k,n}(s)\cdot f_k \tag{3}$$

where

$$w_{k,n}(s)=\frac{\prod_{j=0,\,j\neq k}^{n-1}(\alpha-j+s)}{\prod_{j=0,\,j\neq k}^{n-1}(k-j)} \tag{4}$$

$w_{k,n}(s)$ is a $n-1$ degree polynomial in s.

To obtain the approximations of the function derivatives in x, the $w_{k,n}(s)$ is differentiated with respect to s, as shown:

$$\frac{dP}{dx}=\frac{dP}{ds}\frac{ds}{dx}=\frac{1}{h}\frac{dP}{ds}=\frac{1}{h}\sum_{k=0}^{n-1}\frac{dw_{k,n}(s)}{ds}f_k \tag{5}$$

For the higher order derivatives,

$$\frac{d^vP}{dx^v}=\frac{1}{h^v}\sum_{k=0}^{n-1}w_{k,n}^{(v)}(s)\cdot f_k \tag{6}$$

where

$$w_{k,n}^{(v)}(s)=\frac{d^v w_{k,n}(s)}{ds^v} \tag{7}$$

In the interpolation and approximation of the derivatives (in the developed library), the used polynomials $w_{k,n}^{(v)}(s)$ were calculated for four points ($n = 4$). See Table 1.

Expressing the Lagrange polynomial as a sum of products of f_k and the weight $w_{k,n}(s)$, eq. (2), permits to obtain the derivatives in a straightforward way, eq. (5). Also, calculating the tensor product[31] of the weights $w_{k,n}(s)$, an approximation for a multidimensional regular array can be figured out. For a three-dimensional (3D) grid, for each dimension, the value of s_i is calculated ($s_1 = (x - x_\alpha)/h_1$, $s_2 = (y - y_\alpha)/h_2$), $s_3 = (z - z_\alpha)/h_3$), then the $w_{k_i,n_i}(s_i)$ polynomials are evaluated. These polynomials multiply the function values for each grid point ($f_{k_1k_2k_3}$):

$$P=\sum_{k_1=0}^{n_1-1}\sum_{k_2=0}^{n_2-1}\sum_{k_3=0}^{n_3-1}w_{k_1,n_1}(s_1)\cdot w_{k_2,n_2}(s_2)\cdot w_{k_3,n_3}(s_3)\cdot f_{k_1k_2k_3} \tag{7}$$

where n_1, n_2, and n_3 are the numbers of points used in the interpolation along each dimension.

Differentiating the resulting polynomial with respect to s_i, the approximation to any of the derivatives, with respect to x, y, z variables, is obtained.

$$\frac{\partial P^{v_1+v_2+v_3}}{\partial x^{v_1}\partial y^{v_2}\partial z^{v_3}}=\frac{1}{h_1^{v_1}h_2^{v_2}h_3^{v_3}}\sum_{k_1=0}^{n_1-1}\sum_{k_2=0}^{n_2-1}\sum_{k_3=0}^{n_3-1}w_{k_1,n_1}^{(v_1)}(s_1)\cdot w_{k_2,n_2}^{(v_2)}(s_2)\cdot w_{k_3,n_3}^{(v_3)}(s_3)\cdot f_{k_1k_2k_3} \tag{8}$$

Equation (8) is equivalent to eq. (7), when $v_1 = v_2 = v_3 = 0$, understanding that $w_{k,n}^{(0)}(s) = w_{k,n}(s)$.

The developed library principally consist in routines that, given the function values in a 3D regular grid, calculate the approximate value of the function, gradient vector, and Hessian matrix, in any space point inside the grid limits. These routines are implementations of eq. (8) with $n_1 = n_2 = n_3 = 4$ (tricubic Lagrange interpolation). Different from trilinear

Table 1. $w_{k,n}^{(v)}(s)$ polynomials defined in eqs. (7) and (4), with $n = 4$ y $\alpha = 1$, and $w_{k,n}^{(0)}(s) = w_{k,n}(s)$.

	$k = 0$	$k = 1$	$k = 2$	$k = 3$
$v = 0$	$(-s^3 + 3s^2 - 2s)/6$	$(s^3 - 2s^2 - s + 2)/2$	$(-s^3 + s^2 + 2s)/2$	$(s^3 - s)/6$
$v = 1$	$(-3s^2 + 6s - 2)/6$	$(3s^2 - 4s - 1)/2$	$(-3s^2 + 2s + 2)/2$	$(3s^2 - 1)/6$
$v = 2$	$-s + 1$	$3s - 2$	$-3s + 1$	s

interpolation,[33] this allows for approximations of second-order derivatives (see Table 1).

Inclined grid

The "regular" term for the grid, means that the space between the points (h_i) is constant along any particular dimension; however, it can be different along the other dimension. Moreover, the grid can be inclined (Fig. 2). This means that, at least one of the angles among the axes of the dimensions is different from 90° (this is usual for the unit cell of many crystalline solids). In this case, we always take one axis parallel to the x axis and another parallel to the xy plane.

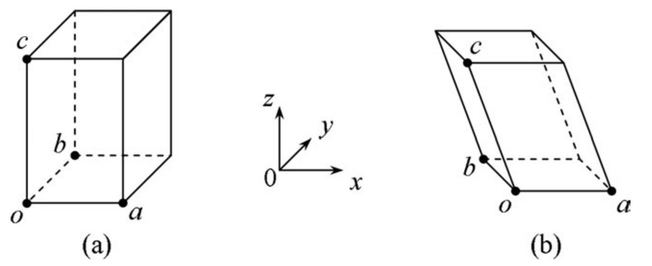


Figure 2. a) No inclined grid, o is situated in the origin, a in the x axis, b in the y axis, and c in the z axis. b) Inclined grid, o is situated in the origin, a in the x axis, b in the xy plane and c could be in any place.

To calculate the evaluated gradient in a space point **r** in an inclined grid, it is necessary to perform two linear transformations,[34] first, the point coordinates in the inclined coordinated system (**r**′) must be calculated, **Ar** = **r**′, where **A** is the matrix that transforms the ordinary (no inclined) coordinates to inclined coordinates. Then, calculating the derivatives, using eq. (8), the gradient vector (**g**′) is determined. Finally, the gradient is transformed back to the ordinary coordinates, multiplying by the transpose of **A**, $\mathbf{g} = \mathbf{A}^T\mathbf{g}'$.

To calculate the Hessian matrix (**H**), it is also necessary to transform the **r** point to the inclined coordinates system. Once **H**′ is calculated (using eq. (8) to get the second-order derivatives), **H**′ is finally transformed back to the ordinary coordinates: $\mathbf{H} = \mathbf{A}^T\mathbf{H}'\mathbf{A}$.

These equations are quite general, when a linear transformation exists from a coordinated system to another, however, in the developed library; the only case considered is when the inclined coordinated system satisfies the given conditions (Fig. 2). In this case, the transform matrix is always an upper triangular matrix.

Approximation to the function logarithm

The electronic charge density has particular characteristics, for example, its value is never negative, and it has the exponential behavior close to the atomic nuclei. The great increment of the density value causes that the interpolated polynomial oscillates producing negative values in the neighborhood of the nuclei (see Fig. 3b). By changing the grid value ($f_{k_1k_2k_3}$) for its logarithm in the previously described interpolation, eq. (7), and finally taking the antilogarithm of the interpolated value (P), it is possible to avoid the oscillation condition.

In the derivative cases, eqs. (9) and (10) are used.

$$\frac{\partial f}{\partial u} = \frac{\partial p}{\partial u} e^p \quad (9)$$

$$\frac{\partial^2 f}{\partial u \partial v} = \left(\frac{\partial^2 p}{\partial u \partial v} + \frac{\partial p}{\partial u}\frac{\partial p}{\partial v} \right) e^p \quad (10)$$

where u and v are x, y, or z, and p is the interpolation of the logarithm. The $\partial p/\partial u$, $\partial p/\partial v$, and $\partial^2 p/\partial u \partial v$ values are the first-order and second-order derivative approximations obtained by the method when the logarithms of the grid values are used.

Methods Requiring the Gradient Vector and the Hessian Matrix

The following methods are used in the topological study and require the gradient vector and the Hessian matrix.

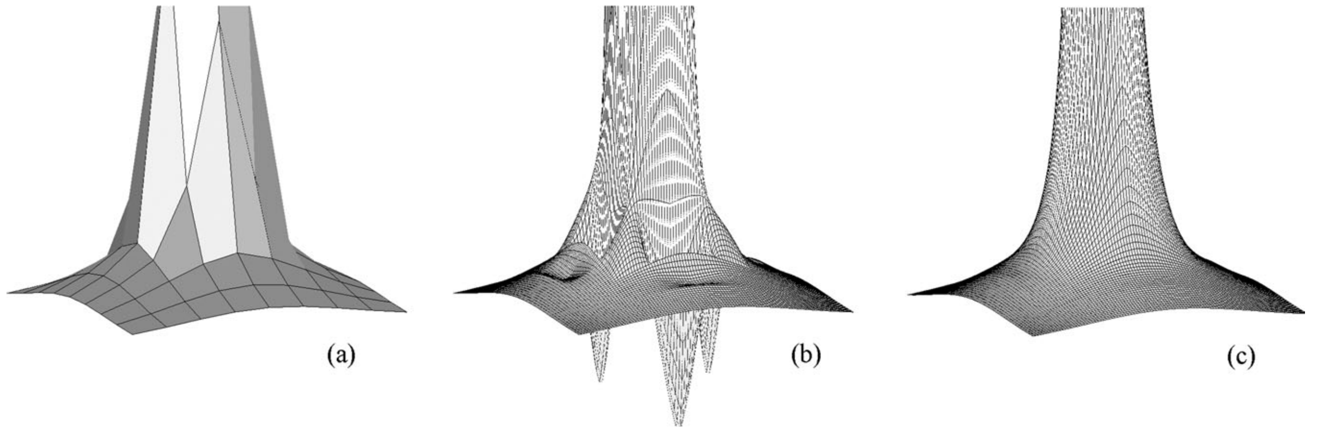


Figure 3. Electronic charge density plots in a plane that contains the sulfur atom of the dibenzotiophene molecule. The plane contains 9 × 9 grid points; the distance between points is 5 × 10^{-12} m. a) Without interpolation. b) Using polynomial interpolation. c) Using logarithm interpolation.

Newton–Raphson method

Due to the fact that the evaluated function gradient at a critical point is equal to the null vector (**0**), the way to calculate the point coordinates is to solve the equation $\nabla\rho(\mathbf{r}_{\text{crit}}) = \mathbf{0}$. An alternative way to solve this equation is to use the Newton–Raphson method.[33] The multidimensional scalar function $\nabla\rho(\mathbf{r})$ evaluated at the point $\mathbf{r} = \mathbf{r}_i + \mathbf{h}$ is expanded in Taylor series:

$$\nabla\rho(\mathbf{r}) = \nabla\rho(\mathbf{r}_i) + \mathbf{H}_i\mathbf{h} + \text{higher order terms} \tag{9}$$

where $\mathbf{H}_i$ is the Hessian matrix $\mathbf{H}(\mathbf{r})$ (the Jacobian of $\nabla\rho$ ($\mathbf{r}$)) valuated at $\mathbf{r}_i$.

By neglecting the higher order terms in the Taylor series, eq. (9), setting $\nabla\rho$ ($\mathbf{r}$) $= \mathbf{0}$ and solving for **h**:

$$\mathbf{h} = -\mathbf{H}_i^{-1}\nabla\rho(\mathbf{r}_i) \tag{10}$$

we get the shift vector (**h**).

If the function $\rho(\mathbf{r})$ is quadratic then the **h** vector starts in the $\mathbf{r}_i$ point and ends in the critical point. In general, the function is not quadratic so a new $\mathbf{r}_{i+1}$ point is always calculated using:

$$\mathbf{r}_{i+1} = \mathbf{r}_i + t\,\mathbf{h} \tag{11}$$

where t is a small value, lower than 1.

The calculation according to eqs. (10) and (11) is iterated until $\nabla\rho(\mathbf{r})$ is equal to the vector **0** or has a small norm. Then, $\mathbf{r}_i$ is a critical point or one very close to it.

In the developed algorithm, the norm of **h** has an upper bound. When $|\mathbf{h}|$ is greater than the bound, t is calculated so that $|t\,\mathbf{h}|$ is equal to the bound (in other case $t = 1$). For each iteration, the bound is decreased by means of a geometric progression, in which the ratio must be lower than 1. Doing this, the large oscillation near to the critical points is avoided, when the gradient has a high norm value in the critical point neighborhood. The maximum number of iterations and the path length are defined by the user.

Fifth-order Cash–Karp Runge–Kutta Method

The gradient path necessary for the molecular graph and interatomic surface construction are solutions of the differential eq. (12).[1]

$$d\mathbf{r}(s)/ds = \nabla\rho(\mathbf{r}(s)) \tag{12}$$

Equation (12) solution is a parametric curve in R^3 which is unique when an initial **r** value is given. A numerical solution is obtained with the fifth-order Cash–Karp Runge–Kutta method [33]. The general form of this method is

$$\mathbf{r}_{n+1} = \mathbf{r}_n + c_1\mathbf{k}_1 + c_2\mathbf{k}_2 + c_3\mathbf{k}_3 + c_4\mathbf{k}_4 + c_5\mathbf{k}_5 + c_6\mathbf{k}_6 \tag{13}$$

where $\mathbf{r}_n = (x_n, y_n, z_n)$ and the $\mathbf{k}_j$ values with a little stepsize h are:

$$\begin{aligned} \mathbf{k}_1 &= h\nabla\rho(\mathbf{r})|_{\mathbf{r}=\mathbf{r}_n} \\ \mathbf{k}_2 &= h\nabla\rho(\mathbf{r})|_{\mathbf{r}=\mathbf{r}_n+b_{21}\mathbf{k}_1} \\ &\dots \\ \mathbf{k}_6 &= h\nabla\rho(\mathbf{r})|_{\mathbf{r}=\mathbf{r}_n+b_{61}\mathbf{k}_1+\cdots+b_{65}\mathbf{k}_5} \end{aligned}$$

The particular values of the various constants (c_j, b_{ij}) can be found in the "*Numerical Recipes*."[33] Choosing the stepsize h as an adequate little value, the set of $\mathbf{r}_n$ points is adjusted closely enough to the gradient path.

Derivative Discontinuities

The derivative discontinuities of piecewise Lagrange polynomials do not allow finding some critical points located on the faces of cube grid. Conversely, the spline interpolation assures the derivative continuities, but it requires an array to store the second derivatives and solving several equation systems involving all grid data.[33] To save calculation time and memory storage, we prefer to use the Lagrange interpolation, instead of spline method and solving the discontinuity problem as explained later.

Four grid points are always taken along each dimension ($4 \times 4 \times 4$ for a 3D grid); so that the point, in which the derivative will be calculated, as far as possible, must be inside the central interval (Fig. 4). This causes a change of interpolation polynomial when

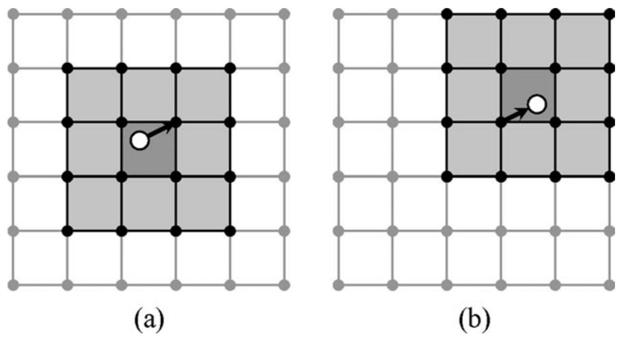


Figure 4. Change of 2D interpolation polynomial. In any point within in the dark gray zone, the values f_{k_1k2k3} used by the interpolation polynomial, eq. (8), correspond to the 16 black dots in the grid. Calculating the gradient or the Hessian matrix in several points starting from the open circle in a) and ending at the open circle in b), the interpolation polynomial is steeply changed when the point reaches the dark gray zone in b).

leaving the central interval. In general, this discontinuity does not represent a serious disadvantage, nevertheless, the Newton–Raphson method can fail when the critical point is exactly on the boundary planes of intervals [with a value of the s_i equal to zero, eqs. (7) and (8)], in which case it is possible that the gradient norm always be greater than the selected convergence criteria.

In the developed library, an option exists that allows continuing with the interpolation polynomial previously used, when the point moves out (not more than 10%) of the interval length in each dimension. This permits finding the critical points that lie in the boundary planes of intervals (in fact, finds a point quite near), without sacrificing the convergence criterion that affects the position of the other critical points.

Library Description

The grid is stored in a structure (of C language), denominated _GRD. Functions that read grid files and return a pointer to a _GRD structure with all the information of the grid (the necessary memory space for this structure is dynamically allocated in the functions) are available. The lagrange3D4grd.c file

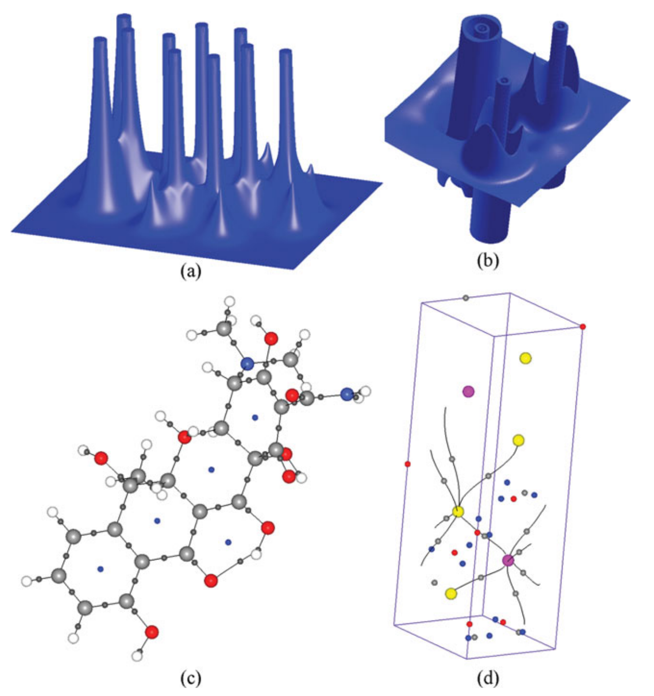


Figure 5. Library application examples: a) Image of the electronic charge density in the plane containing all atoms of a p-nitroaniline molecule. b) Image of the minus Laplacian of the electronic charge density in the plane containing one molybdenum atom and two sulfur atoms of the MoS_2 crystal. c) Bond paths (black lines), and critical points (small gray spheres: bond critical points; blue: ring critical points) of the tetracycline molecule. d) Some bond paths and critical points of the MoS_2 crystal.

contains the read_grd function that reads DMol3 grd files;[35] further description for this function and the _GRD structure are in lagrange3D4grd.h file. GaussianCube.c file contains functions to read grid data in CUBE format from Gaussian[36] and Gamess[37] output files and CHGCARfile.c file the function to read a grid data from Vasp[38] output files.

The C function argument is a character string (**char** * filename) that must contain the name of a *.grd file.

When the _grd_latency global variable has values different from zero it prevents the change of the interpolated polynomial at the interval edges, according to the explanation in Derivative Discontinuities Section.

Functions to calculate the interpolated value, the gradient and Hessian matrix at any space point are described in lagrange3D4grd.h file; also functions to find critical point based on the Newton–Raphson method (described in the Newton–Raphson method Section) and to calculate the points at the gradient path according to the fifth-order Cash–Karp Runge–Kutta method (described in the Fifth-order Cash–Karp Runge–Kutta Method Section).

The library code files can be obtained from the Website: http://alfa.facyt.uc.edu.ve/quimicomp/

Some Examples

The examples shown in Figure 5 are screenshots of graphic windows of some programs that use the described library. The programs were built for Windows XP with MinGW (MinGW is a collection of freely available programming tools, specific header files and import libraries that allow one to produce native Windows® programs Website: http://www.mingw.org/) C compiler and the Dev-C++ (Dev-C++ is a development environment for the C/C++ programming language (for Windows). Free software including a MinGW version Website: http://www.bloodshed.net/devcpp.html) development environment, using OpenGL, Glut, and Glui libraries (Glut and Glui libraries to install it in Dev-C++ can be obtained from the Website: http://www.nigels.com/glt/devpak/).

As an interpolation example, Figure 5a shows the electronic charge density plot on the plane containing all atoms of p-nitroaniline. The calculation was performed using the software Gamess-US[37] at the HF/6-31G(2p,2d) level and the generated grid spacing was 0.1 Bohr. As an example of derivatives calculation with the interpolation of the logarithm, Figure 5b is the graph of the Laplacian of electronic charge density of crystalline MoS_2 on a plane containing one Mo and two S atoms, calculated by density functional theory (DFT) with Perdew-Burke-Ernzerhof (PBE) exchange-correlation functional, and double numerical plus polarization (DNP) basis set using the software DMol^3.[35] Figure 5c is an example of critical points calculation and gradient path for tetracycline (Newton–Raphson and Cash–Karp Runge–Kutta methods, respectively) calculated using the software Gamess-US[37] at the HF/6-31G(2p,2d) level; the grid spacing was 0.05 bohr. In Figure 5d, some bond paths and critical points of the MoS_2 crystal (inclined unit cell) are shown.

For tetracycline, 65 critical points (different from nuclear positions) were determined with a program using this library and also with an analytical electron density method (Multiwfn[26]). The maximal distance, between the calculated critical point position with this method and the Multiwfn program, was 0.0013 Bohr, equivalent to 2.6% of the grid spacing; and 58 of the 65 points have a distance less than 0.0005 Bohr. These differences can be diminished by refining the grid.

Acknowledgments

The authors express thanks to Professor Oscar Valbuena for his help in preparing this manuscript.

Keywords: multidimensional interpolation · QTAIM · C language

How to cite this article: D. Vega, Y. Aray, J. Rodríguez, *J. Comput. Chem.* **2012,** *33,* 2526–2531. DOI: 10.1002/jcc.23083

[1] (a) R. F. W. Bader, Atoms in Molecules-a Quantum Theory; Clarendon Press: Oxford, U.K., **1990**; (b) R. F. W. Bader, *Chem. Rev.* **1991**, *91,* 893.
[2] R. F. W. Bader, *J. Phys. Chem. A* **1998**, *102,* 7314.
[3] R. F. W. Bader, P. L. A. Popelier, T. A. Keith, *Angew. Chem. Int. Ed. Engl.* **1994**, *33,* 620.
[4] P. F. Zou, R. F. W. Bader, *Acta Crystallogr.* **1994**, *A50,* 714.
[5] M. E. Eberhart, *Can. J. Chem. Soc.* **1996**, *74,* 1229.
[6] M. A. Pendás, A. Costales, V. Luaña, *Phys. Rev. B* **1997**, *55,* 4275.
[7] P. L. A. Popelier, Atoms in Molecules—An Introduction; Prentice Hall: Harlow-England, **2000**.
[8] R. J. Gillespie, P. L. A. Popelier, Chemical Bonding and Molecular Geometry: From Lewis to Electrón Densities; Oxford University Press: New York, Oxford, **2001**.

[9] Y. V. Ivanov, E. L. Belokoneva, J. Protas, N. K. Hansen, V. G. Tirelson, *Acta Crystallogr. B* **1998**, *54,* 774.
[10] T. S. Koritsanszky, P. Coppens, *Chem. Rev.* **2001**, *101,* 1583.
[11] A. Fischer, D. Tiana, W. Scherer, K. Batke, G. Eickerling, H. Svendsen, N. Bindzus, B. B. Iversen. *J. Phys. Chem. A,* **2011**, *115,* 13061.
[12] L. J. Farrugia, P. Macchi. *Struct. Bond* **2012**, 1, *146*, 127.
[13] R. Boese, N. Niederprüm, D. Bläser, A. Maulitz, M. Y. Antipin, P. R. Mallinson, *J. Phys. Chem. B* **1997**, *101,* 5794.
[14] P. Munshi, T. N. Guru Row, *Crystallogr. Rev.* **2005**, *11,* 199.
[15] C. J. Mei, K. E. Edgecombe, V. H. Smith, A. Heilingbrunner, *Int. J. Quantum Chem.* **1993**, *48,* 287.
[16] G. H. Grosch, K. J. Range, *J. Alloys Compd.* **1996**, *233,* 39.
[17] M. Knecht, H. Ebert, W. Bensch, *J. Alloys Compd.* **1997**, *246,* 166.
[18] C. Gatti, V. R. Saunders, C. Roetti, *J. Chem. Phys.* **1994**, *101,* 10686.
[19] J. A. Platts, S. T. Howard, *J. Chem. Phys.* **1996**, *105,* 4668.
[20] (a) P. L. A. Popelier, *Comp. Phys. Comm.* **1996**, *93,* 212; (b) P. L. A. Popelier, *Comp. Phys. Comm.* **1998**, *108,* 180.
[21] C. Gatti, An Electron Density Topological Program for Systems Periodic in N ($N =$; 0–3) Dimensions; CNR-ISTM: Milano, **1999**.
[22] AIM2000: F. W. Biegler-König, J. Schönbohm, D. Bayles, Website: http://gauss.fh-bielefeld.de/aim2000. Accessed on February 15, 2012.
[23] M. Barzaghi:PAMoC (Version 2002.0), Online User's Manual; CNR-ISTM, Institute of Molecular Science and Technologies: Milano, Italy, 2002. Website: http://www.istm.cnr.it/~barz/pamoc/. Accessed on February 15, 2012.
[24] AIMAll (Version 12.06.03), Todd A. Keith, TK Gristmill Software, Overland Park KS, USA, 2012, Website: http://aim.tkgristmill.com. Accessed on February 15, 2012.
[25] Aimpac: http://www.chemistry.mcmaster.ca/aimpac/imagemap/imagemap.htm. Accessed on February 15, 2012.
[26] Multiwfn: T. Lu, F. Chen, *J. Comp. Chem.* **2012**, *33,* 580.
[27] M. Kohout, DGrid 4.6, User's Guide, Max Planck Institute for Chemical Physics of Solids, Dresden, Germany, 2011. Website: http://www.cpfs.mpg.de/~kohout/Documents/DGrid-4.6.pdf. Accessed on February 15, 2012.
[28] Critic:A. Otero-de-la-Roza, M. A. Blanco, A. Martín Pendás, V. Luaña, *Comp. Phys. Comm.* **2009**, *180,* 157.
[29] Y. Aray, J. Rodriguez, D. Vega, In The Quantum Theory of Atoms in Molecules- From Solid State to DNA and Drug Design; C. Matta, R. Boyd, Eds. Wiley-VCH, **2007**; pp 231–256.
[30] Y. Aray, J. Rodríguez, D. Vega, S. Coll, E. Rodríguez-Arias, F. Rosillo, *J. Phys. Chem. B* **2002**, *106,* 13242.; (b) Y. Aray, M. Marquez, J. Rodríguez, S. Coll, Y. Simón-Manso, C. Gonzalez, D. Weitz, *J. Phys. Chem. B* **2003**, *107,* 8946.; (c) Y. Aray, M. Marquez, J. Rodríguez, D. Vega, Y. Simón--Manso, S. Coll, C. Gonzalez, D. Weitz, *J. Phys. Chem. B* **2004**, *108,* 2418.; (d) Y. Aray, J. Rodríguez, S. Coll, E. Rodríguez-Arias, D. Vega, *J. Phys. Chem. B* **2005**, *109,* 23564.; (e) Y. Aray, A. Vidal, J. Rodriguez, M. Grillo, D. Vega, D. Coll *J. Phys. Chem. C* **2009**, *113,* 19545.
[31] D. Kincaid, W. Cheney, Numerical Analysis; Brooks/Cole Publishing Company, California, **1991**.
[32] C. Katan, P. Rabiller, C. Lecomte, M. Guezo, V. Oisona, M. Souhassoub, *J. Appl. Cryst.* **2003**, *36,* 65.
[33] W. Press, S. Teukolsky, W. Vetterling, B. Flannery, Numerical Recipes in C; Cambridge University Press, New York, **1992**.
[34] D. Vega, Modelaje de la adsorción de moléculas pequeñas sobre superficies metálicas, Tesis Doctoral; Instituto Venezolano de Investigaciones Científicas: Caracas, Venezuela, **2004**; Apéndice B.
[35] DMol^3, User Guide, Release 3.2; Accelrys Software Inc., **2005**.
[36] GAUSSIAN 09 user's reference, 2011: http://www.gaussian.com/g_tech/g_ur/u_cubegen.htm. Accessed on February 15, 2012.
[37] M. S. Gordon, M. W. Schmidt, In Theory and Applications of Computational Chemistry: the first forty years; C. E. Dykstra, G. Frenking, K. S. Kim, G. E. Scuseria, Eds.; Elsevier: Amsterdam, **2005**; pp. 1167–1189.
[38] VASP the GUIDE, Universität Wien, Austria, 2011, Website: http://cms.mpi.univie.ac.at/vasp/vasp/vasp.html.

Received: 17 February 2012
Revised: 12 June 2012
Accepted: 12 July 2012
Published online on 2 August 2012