Axl, a geometric modeler for semi-algebraic shapes

Emmanouil Christoforou, Angelos Mantzaflaris, Bernard Mourrain, Julien Wintz

Axl, a geometric modeler for semi-algebraic shapes

E. Christoforou[1,2], A. Mantzaflaris[3], B. Mourrain[1] and J. Wintz[1]

[1] Univ. Côte d'Azur, Inria Sophia-Antipolis - Méditerranée, France
firstname.lastname@inria.fr,
http://team.inria.fr/aromath/
[2] National and Kapodistrian University of Athens, Greece
echristo@di.uoa.gr,
[3] Johannes Kepler University, Linz, Austria
angelos.mantzaflaris@oeaw.ac.at

Abstract. We describe the algebraic-geometric modeling platform Axl, which provides tools for the manipulation, computation and visualisation of semi-algebraic models. This includes meshes, basic geometric objects such as spheres, cylinders, cones, ellipsoids, torus, piecewise polynomial parameterisations of curves, surfaces or volumes such as b-spline parameterisations, as well as algebraic curves and surfaces defined by polynomial equations. Moreover, Axl provides algorithms for processing these geometric representations, such as computing intersection loci (points, curves) of parametric models, singularities of algebraic curves or surfaces, certified topology of curves and surfaces, etc.
We present its main features and describe its generic extension mechanism, which allows one to define new data types and new processes on the data, which benefit from automatic visualisation and interaction facilities. The application capacities of the software are illustrated by short descriptions of plugins on algebraic curves and surfaces and on splines for Isogeometric Analysis.

Keywords: semi-algebraic model, isogeometric analysis, b-splines, algebraic surface, algebraic-geometric computation, generic programming

1 Introduction

Geometric modeling aims at providing shape descriptions and at developing computational tools for processing the models. It has strong interactions with other application domains such as graphical rendering and visualisation, Computer Aided Design and Computer Aided Manufacturing, numerical simulation, etc.

Many of the models which are used are semi-algebraic sets. Meshes, classically used to approximate shapes, are piecewise linear models. b-splines or NURBS curves and surfaces used in CAD-CAM are the images of piecewise polynomial or rational parametrisation maps. Natural quadrics such as spheres, ellipsoids, cylinders, cones or higher order surfaces such as torus are algebraic surfaces defined by polynomial equations. Semi-algebraic models of order higher than one have interesting properties of approximation and regularity, allowing to construct high quality shape representations.

However, currently very few software are able to manipulate these different types of semi-algebraic sets. Software like MESHLAB are PARAVIEW, propose tools for visualization and computations with meshes. BLENDER or RHINO allow one to manipulate b-spline parametric objects. Software like SURF are able to render algebraic surfaces, but does not provide facilities to compute with them.

The goal of the AXL development project (`axl.inria.fr`) is to provide tools for the manipulation, computation and visualisation of semi-algebraic models of higher order. This includes meshes, basic algebraic objects, b-spline parameterisations of curves, surfaces or volumes and semi-algebraic sets defined by polynomial equations. Additionally, AXL provides algorithms to process these geometric representations such as computing intersection points or curves of parametric models, singularities of algebraic curves or surfaces, certified topology of curves and surfaces, etc.

To cope with the versatility of shape representations, AXL integrates a generic extension mechanism, which allows one to define new data types and new processes on this data. As soon as these new instances are constructed, visualisation and interaction facilities are provided essentially automatically. Via the production of dedicated plugins, external tools can be easily embedded, tested and demonstrated in this framework.

In Section 2, we describe the main feature of AXL platform. In Section 3, we describe the design of the code and its extension mechanism. In Section 4, we present applications, with a short description of plugins, respectively, on algebraic curves and surfaces and on splines for Isogeometric Analysis.

2 Functionalities

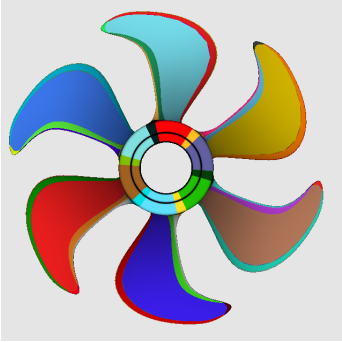
AXL software provides different types of semi-algebraic representations of shapes used in geometric modeling, such as the image of piecewise polynomial or rational maps from a bounded parameter domain into $\mathbb{R}^2$ or $\mathbb{R}^3$ or solutions of polynomial equations or geometric constructions on these objects.

Basic geometric objects such as points, segments, circular arcs, planes, spheres, cylinders, cones, ellipsoids or tori are available in the AXL library. These types correspond to specific classes with compact representations (`axlPoint`, `axlLine`, `axlSphere`, ...). For instance, an ellipsoid is represented by a center point and 3 orthogonal vectors defining its principal axis. The coordinates of these objects are stored as floating point numbers (`double` precision in the IEEE 754 standard). These objects can be edited interactively in two ways: either graphically via widget actors in the view window or through the object inspector panel of the application by changing directly the value of the numerical data representing the object. More complex objects such as meshes (`axlMesh`) are also available; they are represented as arrays of points, edges and faces with an arbitrary number of vertices. Normals or other attributes can be attached to the points.

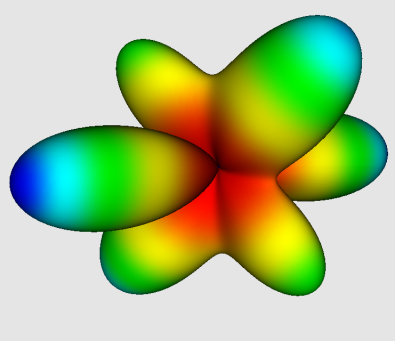
b-spline and NURBS (Non-Uniform Rational B-Spline) parametrisations of curves, surfaces and volumes are provided by a specialized plugin `bsplinetools`, which is based on the library GoTools[4] developed by SINTEF. Such parametrisation maps are represented by arrays of control points, and knot sequences. Abstract classes (`axlAbstractCurveBSpline`, `axlAbstractSurfaceBSpline`, ...) specify the available methods, independently of the internal representation of the data. Dedicated widget actors allow one to edit dynamically these objects by changing graphically the control points, using the methods of the abstract interface classes.

Algebraic curves and surfaces defined by polynomial equations are implemented in the plugin `semialgebraictools`. They are represented by arrays of multivariate polynomials, with exact or approximate coefficients. They are embedded in the Axl framework, through abstract interface classes (`axlAbstractCurveAlgebraic`, `axlAbstractSurfaceAlgebraic`).

Geometric types of the Axl library derive from the generic class `axlAbstractData` and share color, transparency and shader attributes. Additionally, field attributes can be attached to the geometric objects. They can be scalar fields, visualized by a color map or vector fields visualized by small arrows. Their representations can depend on the type of the supporting geometric object. They can be discrete values at the vertices of a mesh, functions of the parameters on a parametric curve or surface or functions of the spatial coordinates of the points on the geometric object.



(a) A collection of b-spline surfaces with different color attributes.



(b) A mesh with a spatial scalar field visualized by a color map.

To be able to exchange data, each data type in Axl is equipped with a reader and writer class. The writer class writes the information that define the objects in a XML-like format, which is simple to access and exploit. The reader class reads the XML text and constructs the corresponding Axl object. Here is an example of format for an ellipsoid, with name and color attributes:

```
<ellipsoid name="E0" color="160 0 32 1">
  <center>0.417312 0.466944 0.729041</center>
  <semix>0.603441 0.0735849 -0.425506</semix>
```

[4] https://www.sintef.no/projectweb/geometry-toolkits/gotools/

```
  <semiy>-0.00639521 0.395524 0.0593304</semiy>
  <semiz>0.116345 -0.022291 0.161143</semiz>
</ellipsoid>
```

The library Axl also provides facilities to run computation on geometric objects. This is implemented via the concept of process, corresponding to the abstract class `axlAbstractProcess`. A process class has input data, a `run` method and output data. This construction allows one to run interactively specific computations, implemented in the `run` command, on selected data and to view the result of the computation. Processes can be selected in the tool inspector panel of the application user interface and executed on the data selected from the user interface. Processes can also be stored in XML-format via their writer class and executed on data via their reader class.

A special type of process allows one to update interactively the result. It involves the, so called, dynamic data class (`axlDataDynamic`), consisting of input data, a process and output data. When the input data is modified, the output of the dynamic data is recomputed. Output data can be the input data of other dynamic objects. This allows one to develop complex constructions of geometric objects, which are updated interactively when some of their components are modified.

Another type of process allows one to create animations in Axl. These consist of input data and a `run` command, which transforms the input data according to a time parameter and visualizes the transformed data according to this time parameter.

3 A generic platform for geometric computation

The Axl application is developed in C++, depending on the *Qt*, *VTK* [12] and *dtk* [13] libraries. *Qt* is a cross-platform application development framework for applications that is compatible with various software and hardware platforms and is used for the graphical user interface of Axl. The *dtk* library is a meta-platform for modular scientific platform development, which provides generic tools for data, processes and views. The visualization plugin of Axl, `axlVtkView`, is based on *VTK*, which is an open source software system for 3D computer graphics, image processing, and visualization.

Using polymorphism, which can be either static or dynamic, Axl is a modular platform, in the sense that it formalizes abstract concepts such as data, processes or views. The latter are then virtual abstractions which can be specialized through plugins, dynamic libraries loaded at runtime fulfilling an abstraction specification. The extensive use of design patterns, such as factory, template method etc. makes it easy to specify the actual behavior of an algorithm by selecting a combination of processes acting on various data representations via their abstraction.

As a matter of fact, Axl provides a generic interface for geometric concepts that which curves, surfaces and volumes with different representations (that can

be implicit, explicit or piecewise linear) and processes such as differentiation, intersection, arrangements, singular points computation etc. Instances of these concepts are commonly implemented by third-party libraries, using diverse algorithms under the hood, that are, in principle, conflicting one another, making their combination problematic. This problem is tackled by the abstraction level of Axl, ensuring the consistency of the different implementations.

Let us describe some of them starting with the virtual hierarchy of data, then some process and how they are combined in order to implement an algorithm. We will not focus on the view concept implemented in the plugin `axlVtkView` based on VTK, which basically renders the meshes output by the converters of the semi-algebraic models and instantiates the graphical actors.

Starting from the virtual hierarchy of data in Axl, in a simple case they inherit from the class `axlAbstractData`, described in previous section, which inherit from `dtkAbstractData`. Figures 2a, 2b show the inheritance of `axlPoint` and `axlLine` with more complex hierarchy. Likewise, Axl processes inherit from class `axlAbstractProcess`, also described previously, that inherit from `dtkAbstractProcess` (Figure 2c, `axlIntersection` example). The use of the abstract classes of data as the processes default input and output, allows processes to handle multiple data-types when possible. In particular, the different data-types and processes acquire a common input and output, thus they can be easily combined or changed in their algorithmic implementation.

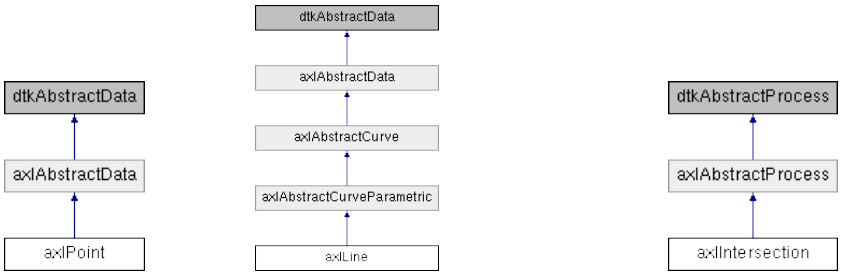
Axl provides also the tools to extend data-types and processes. New data-types, which inherit from `axlAbstractData`, are created having their own reader, writer, creator, converter etc., by inheriting the corresponding abstract class (`axlAbstractDataReader`, `axlAbstractDataWriter`, etc), in order to be properly integrated and functional in Axl. Also, new processes can be implemented using their abstraction (`axlAbstractProcess`) by defining the input data, a run method and the output data. The new data-types and processes can be used in Axl as plugins, by creating new packages.

4 Applications

In this section we describe two application plugins, that tightly integrate into Axl tools for real algebraic curves and surfaces and for isogeometric analysis.

4.1 Topology of real algebraic sets

The Axl framework has been used to develop a plugin called `semialgebraictools`, dedicated to the topology analysis of algebraic curves and surfaces and to the computation of arrangements of such objects. Algebraic curves and surfaces are defined as the real solutions of polynomial equations. In this implementation, planar



(a) Inheritance of data. `axlPoint` is a point defined by 3 coordinates.

(b) Inheritance of data. `axlLine` is a line (or segment) defined by a starting and an ending point (`axlPoints`) deriving from the interface abstract class of parametric curve.

(c) Inheritance of processes. `axlIntersection` intersects two lines and returns the point of intersection.

Fig. 2: Axl inheritance

algebraic curves defined by one equation, curves in $\mathbb{R}^3$ defined by two equations and surfaces in $\mathbb{R}^3$ defined by one equation are considered. The polynomials are represented in the Bernstein basis associated to a given domain (that is, an axis aligned box of dimension 2 or 3) by a matrix or a tensor of control coefficients. The topology of the algebraic objects is analyzed in this region. Subdivision methods are used to compute a mesh approximation of the algebraic set, which is topologically certified. The subdivision steps consists in splitting the domain in one direction and in computing the Bernstein basis representation on each subdomain, using de Casteljau algorithm. Regularity criteria are used to determine whether the topology of the algebraic object can be determined from its intersection points with the edges (or faces) of the box [1].

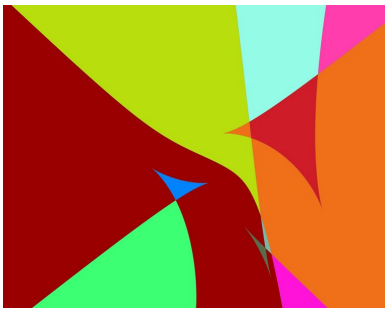
The computation is performed on the tensor representation in the Bernstein bases with lower and upper approximate coefficient bounds (`double` type of the IEEE 754 standard). By choosing adequately the rounding mode during the computation, the exact value of the coefficients is guaranteed to stay between the computed lower and upper bounds [11].

New data types encoding the bounding box domain and the polynomial equations have been implemented in this plugin. The visualization of the algebraic sets is performed by a converter class, which computes a mesh from the polynomial equations by subdivision methods.

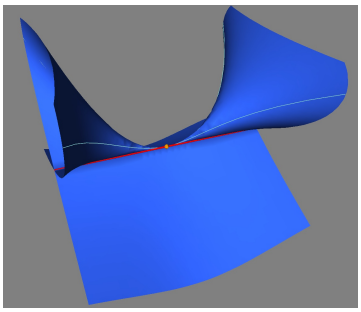
These subdivision methods have been used to compute the topology of algebraic curves [2, 9] and algebraic surfaces [3], arrangements of curves [4], semi-algebraic sets [10] and Voronoï diagrams of curved objects [5], see Figure 3.

4.2 Modeling and simulation

Isogeometric Analysis is a new, innovative numerical technique that generalizes the Finite Element Method and uses splines or NURBS, normally used in Computer



(a) The regions defined by an algebraic curve with singular points of degree 28, which contains the medial axis of two ellipses.



(b) An algebraic surface with a singular curve (in red) and a special singular point (in orange) defining its Whitney stratification and the apparent contour (in light blue).

Fig. 3: Views of an algebraic curve and an algebraic surface.

Aided Design, for both representing the geometry of the computational (physical) domain and for approximating the solution of the considered partial differential equation. In this paragraph we present the related AXL plugin based on the G+SMO library (`http://www.gs.jku.at/gismo`).

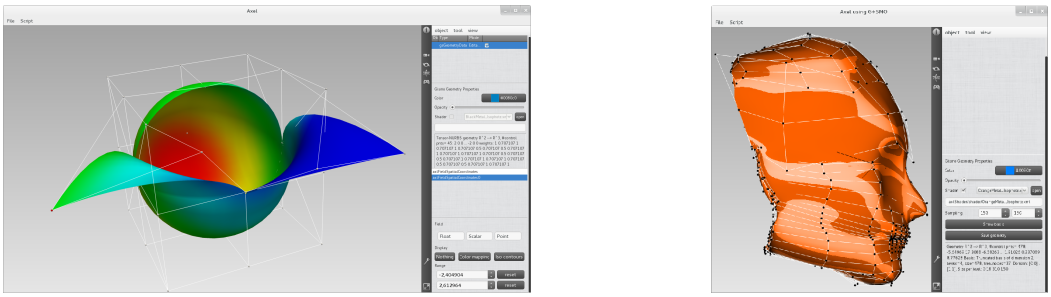
G+SMO is an open-source, object-oriented C++ library for isogeometric analysis [7, 8]. The library makes use of object polymorphism and inheritance techniques in order to support a variety of different discretisation bases, namely b-spline, Bernstein, NURBS bases, hierarchical and truncated hierarchical b-spline bases of arbitrary polynomial order. The implementation of basis functions and geometries is dimension-independent, that is, curves, surfaces, volumes, bulks (in 4D) and other high–dimensional objects are instances of code templated with respect to the parameter domain dimension.

Three general guidelines have been set for the development process. Firstly, we promote both efficiency and ease of use; secondly, we focus on code quality and cross-platform compatibility and, thirdly, we encourage the exploration of new strategies, better suited for isogeometric analysis before adopting existing finite element practices.

The library is partitioned into modules that implement different functionalities. A basic module that is available is the *NURBS module*, which provides a dimension independent implementation of classical tensor-product b-splines and their rational counterpart. On top of the NURBS module we implemented the *hierarchical splines module* [6]. The functionalities can be used seamlessly in AXL via our plugin; Figure 4 shows two instances of its use.

References

1. Alberti, L., Mourrain, B.: Regularity criteria for the topology of algebraic curves and surfaces. In: Martin, R., Sabin, M., Winkler, J. (eds.) Mathematics of Surfaces XII, LNCS, vol. 4647, pp. 1–28.



(a) Peeling of a NURBS sphere interactively in Axl. The original sphere is represented by a biquadratic tensor-product NURBS surface. Editing triggers evaluation of both the surface and the scalar field on a grid of points in real-time.

(b) A THB-spline model in Axl; note the accumulation of control point near the mouth region. A shader using isophotes is applied.

Fig. 4: Two snapshots of the G+Smo plugin.

Springer (2007)
2. Alberti, L., Mourrain, B.: Visualisation of implicit algebraic curves. pp. 303–312. Pacific Conference on Computer Graphics and Applications, IEEE Computer Society, Lahaina, Maui, Hawaii, United States (Oct 2007)
3. Alberti, L., Mourrain, B., Técourt, J.P.: Isotopic triangulation of a real algebraic surface. Journal of Symbolic Computation 44(9), 1291–1310 (Feb 2009)
4. Alberti, L., Mourrain, B., Wintz, J.: Topology and arrangement computation of semi-algebraic planar curves. Computer Aided Geom. Design 25(8), 631–651 (2008)
5. Emiris, I., Mantzaflaris, A., Mourrain, B.: Voronoi diagrams of algebraic distance fields. Computer-aided Design 45(2), 511 – 516 (2013)
6. Giannelli, C., Juettler, B., Kleiss, S.K., Mantzaflaris, A., Simeon, B., Speh, J.: THB-splines: An effective mathematical technology for adaptive refinement in geometric design and isogeometric analysis. Computer Methods in Applied Mechanics and Engineering 299, 337 – 365 (2016)
7. Juettler, B., Langer, U., Mantzaflaris, A., Moore, S., Zulehner, W.: Geometry + simulation modules: Implementing isogeometric analysis. Proc. Appl. Math. Mech. 14(1), 961–962 (2014)
8. Langer, U., Mantzaflaris, A., Moore, S., Toulopoulos, I.: Multipatch discontinuous Galerkin isogeometric analysis. In: Isogeometric Analysis and Applications, pp. 1–32. Lecture Notes in Computational Science and Engineering, Springer (2015)
9. Liang, C., Mourrain, B., Pavone, J.P.: Subdivision Methods for the Topology of 2d and 3d Implicit Curves. In: Juetller, B., Piene, R. (eds.) Geometric Modeling and Algebraic Geometry, pp. 199–214. Springer (2007)
10. Mantzaflaris, A., Mourrain, B.: A subdivision approach to planar semi-algebraic sets. In: Advances in Geometric Modeling and Processing, Lecture Notes in Computer Science, vol. 6130, pp. 104–123. Springer (2010)
11. Mourrain, B., Pavone, J.P.: Subdivision methods for solving polynomial equations. Journal of Symbolic Computation 44(3), 292–306 (2009)
12. Schroeder, W., Martin, K., Lorensen, B.: The Visualization Toolkit (4th ed.). Kitware (2006)
13. Wintz, J., Kloczko, T., Niclausse, N., Rey, D.: dtk - A metaplatform for scientific software development. ERCIM News 2012(88) (2012), http://ercim-news.ercim.eu/en88/ri/dtk-a-metaplatform-for-scientific-software-development