Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

Particle-based and meshless methods with Aboria

Martin Robinson [a,*], Maria Bruna [b]

[a] Department of Computer Science, University of Oxford, Wolfson Building, Parks Rd, Oxford OX1 3QD, United Kingdom
[b] Mathematical Institute, University of Oxford, Radcliffe Observatory Quarter, Woodstock Road, Oxford OX2 6GG, United Kingdom

A R T I C L E   I N F O

Article history:
Received 31 March 2017
Received in revised form 3 July 2017
Accepted 5 July 2017

Keywords:
Particle-based numerical methods
Meshless
C++

A B S T R A C T

Aboria is a powerful and flexible C++ library for the implementation of particle-based numerical methods. The particles in such methods can represent actual particles (e.g. Molecular Dynamics) or abstract particles used to discretise a continuous function over a domain (e.g. Radial Basis Functions). Aboria provides a particle container, compatible with the Standard Template Library, spatial search data structures, and a Domain Specific Language to specify non-linear operators on the particle set. This paper gives an overview of Aboria's design, an example of use, and a performance benchmark.

Code metadata

Current code version	v0.4
Permanent link to code/repository used for this code version	http://github.com/ElsevierSoftwareX/SOFTX-D-17-00029
Legal Code Licence	BSD 3-Clause Licence
Code versioning system used	git
Software code languages, tools, and services used	C++
Compilation requirements, operating environments & dependencies	Tested on Ubuntu 14.04 LTS with the GCC compiler (version 5.4.1), and Clang compiler (version 3.8.0). Third-party library dependencies: Boost, Eigen (optional), VTK (optional)
If available Link to developer documentation/manual	https://martinjrobins.github.io/Aboria
Support email for questions	martin.robinson@cs.ox.ac.uk

1. Motivation and significance

Aboria is a C++ library that supports the implementation of particle-based numerical methods, which we define as having three key properties:

1. There is a set of N particles that have positions within a hypercube of d dimensions, with each dimension being either periodic or non-periodic.
2. The method can be described in terms of non-linear operators on the N particle positions and/or other variables associated to these particles.
3. These operators are defined solely by the particle positions and variables, and typically take the form of interactions between pairs of closely spaced particles (i.e. neighbourhood interactions). There are no pre-defined connections between the particles.

This definition covers a wide variety of popular methods where particles are used to represent either physical particles or a discretisation of a continuous function. In the first category are methods such as Molecular Dynamics [1], Brownian and Langevin Dynamics [2]. The second category includes methods like Smoothed Particle Hydrodynamics (SPH) [3], Radial Basis Functions (RBF) for function interpolation and solution of Partial Differential Equations (PDEs) [4–6], and Gaussian Processes in Machine Learning [7].

To date, a large collection of software has been developed to implement these methods. Generally, each software package focuses on one or at most two methods. Molecular and Langevin Dynamics are well-served by packages such as GROMACS [8], LAMMPS [9], ESPResSo [10] or OpenMM [11]. SPHysics [12] is one of the best

* Corresponding author.
E-mail address: martin.robinson@cs.ox.ac.uk (M. Robinson).

http://dx.doi.org/10.1016/j.softx.2017.07.002

known SPH solvers. There exists no large-scale package for RBF methods, but these can be implemented in Matlab [13], and are available as routines in packages such as SciPy [14].

The software listed above represents a considerable investment of time and money. The computational requirements of particle-based methods, such as the efficient calculation of interactions between particles, are challenging to implement in a way that scales well with the number of particles N, uniform and non-uniform particle distributions, different spatial dimensions and periodicity. Yet, to date there does not exist a general purpose library that can be used to implement these low-level routines, and so they are reimplemented again and again in each software package.

The situation with particle-based methods contrasts with mesh-based methods such as Finite Difference, Finite Volume or Finite Element Methods. These methods are supported by linear algebra libraries based on the BLAS [15] specifications, and today it would be very strange to implement a mesh-based method without using BLAS libraries. In general, particle-based methods cannot take advantage of linear algebra libraries, which are good for static nodes with connections that do not change during the simulation, as opposed to dynamic particles whose interaction lists are variable over the course of the simulation.

Aboria aims to replicate the success of linear algebra libraries by providing a general purpose library that can be used to support the implementation of particle-based numerical methods. The idea is to give the user complete control to define of operators on the particle set, while implementing efficiently the difficult algorithmic aspects of particle-based methods, such as neighbourhood searches and fast summation algorithms. However, even at this level it is not a one-fits-all situation and Aboria is designed to allow users to choose specific algorithms that are best suited to the particular application. For example, calculating neighbourhood interactions for a uniform particle distribution is best done using a regular cell-list data structure, while for a highly non-uniform particle distribution a tree data structure like a k–d tree might be preferable [16]. For neighbourhood interactions that are zero beyond a certain radius, a radial search is the best algorithm to obtain interacting particle pairs, while for interactions that are amenable to analytical spherical expansions, the fast multipole method is an efficient fast summation algorithm [17].

1.1. Software capabilities

The online documentation provides a set of example programs to illustrate Aboria's capabilities. These include:

Molecular Dynamics Simulation of N Newtonian point particles interacting via a linear spring force.

Brownian Dynamics Simulation of N Brownian point particles moving through a set of fixed reflecting spheres that act as obstacles.

Discrete Element Model (DEM) Simulation of N granular particles with drag and gravity, interacting via a linear spring force with varying resting lengths.

Smoothed Particle Hydrodynamics (SPH) Simulation of a water column in hydrostatic equilibrium. SPH discretises the Navier–Stokes equations using radial interpolation kernels defined over a given particle set. See [18] for more details.

Radial Basis Function (RBF) Interpolation Computation of an RBF interpolant of a two-dimensional function using a multiquadric basis function. The linear algebra library Eigen [19] is used to solve the resulting set of linear equations.

Kansa Method for PDEs Application of RBFs (using Gaussian basis functions) to solve the Poisson equation in a square two dimensional domain with Dirichlet boundary conditions.

2. Software description

2.1. Software architecture

The high-level design of Aboria consists of three separate and complimentary abstraction levels (see Fig. 1). Aboria Level 1 contains basic data-structures that implement a particle set container, with associated spatial search capabilities. Level 2 contains efficient algorithms for particle-based methods, such as neighbour searches and fast summation methods. Level 3 implements a Domain Specific Language (DSL) for specifying nonlinear operators on the set of particles. While Levels 1 and 2 provide useful functionality for particle-based methods, the purpose of Level 3 is to tie together this functionality and to provide a easy-to-use interface that ensures that the capabilities of Levels 1 and 2 are used in the best possible way.

2.2. Software functionalities

2.2.1. Aboria Level 1

Aboria Level 1 implements a particle container class that holds the particle data (Fig. 2). This class is based on the Standard Template Library (STL) vector class (Level 0 in Fig. 1), which serves as the lowest-level data container. Three variables are stored by default: `position`, `id` and `alive`, representing respectively the particle's spatial position, unique identification number and a flag to indicate if the particle is marked for deletion. The user can specify the spatial dimension d ($d = 1$ in the example in Fig. 2), as well as any additional variables attached to each particle (an additional variable `velocity` is used in Fig. 2). The Level 1 particle set container will combine multiple Level 0 vectors to form a single data structure.

This particle set container generally follows the STL specification with its own iterators and traits (see Fig. 2). It supports operations to add particles (the STL `push_back` member function), remove particles (`erase`), and can return a single particle given an index i (`operator[]`). This index operation returns a lightweight type containing references to the corresponding index in the set of zipped Level 0 vectors. Individual variables can be obtained from this lightweight type via `get` functions provided by Aboria.

Level 1 also includes spatial search data structures, which can be used for fast neighbour searches throughout a hypercube domain with periodic or non-periodic boundaries. The particle set container interacts with the spatial search data structures to embed the particles within the domain, ensuring that the two data structures are kept synchronised while still allowing for updates to the particle positions.

The current version of the code implements only cell-list search data structures, which divide the domain into a square lattice of hypercube cells, each containing zero or more particles (see Fig. 3). Two cell-list implementations are provided, one which supports insertion of particles in parallel by reordering the particles in the set, and the other which only supports serial insertion. The latter is faster for serial use, and for when particles move rapidly relative to each other (e.g. Brownian dynamics). Both data structures support parallel queries. In future versions of Aboria a k–d tree data structure will also be added.

2.2.2. Aboria Level 2

Aboria Level 2 implements efficient search and fast summation algorithms using the particle set and spatial search data structures in Level 1. Currently Level 2 includes a box search algorithm around a given spatial position $\mathbf{r}_s$. This calculates which cell $\mathbf{r}_s$ is in and searches within that cell and its adjacent cells for potential neighbour particles within the given search box (see Fig. 3). This is useful for particle interactions that are zero beyond a certain radius,